

**Mejora de la Eficiencia en  
Sistemas de Atención al Ciudadano  
mediante IA:  
*Ayuda Inteligente, Detección  
Automática de Errores y Consultas  
Personalizadas***

Autora: Amparo Valenzuela Juan

Tutor: Óscar Corcho

Curso 2024-2025

## Tabla de contenido

Parte 1	7
1. Introducción	7
1.1 Motivación	9
1.2 Objetivos	9
1.3 Estado del arte	10
1.3.1 Enfoques y estudios relacionados	11
Parte 2	14
2. Modelo Semántico y Arquitectura del Sistema basado en Ontología	14
2.1 Creación de la Ontología “Atención al Ciudadano” (ontoAC)	14
2.1.1 Especificación de requisitos	15
2.1.2 Conceptualización	30
2.1.3 Implementación	31
2.1.4 Evaluación	33
2.2 Arquitectura del Sistema basado en Ontología	51
2.2.1 Componentes principales	51
2.2.2 Modelo de datos en Oracle	54
2.2.3 Flujo de interacción	55
2.2.4 Comunicación entre módulos	56
2.2.5 Lógica de diseño	57
2.2.6 Fundamentos del uso exclusivo de Ontop como motor SPARQL	58
2.2.7 Elección de OpenAI para la generación de consultas SPARQL	59
Parte 3	61
3. Desarrollo de la Ayuda Inteligente	61
3.1 Metodología	61
3.2 Implementación del Botón de Ayuda Inteligente	64
3.2.1 Arquitectura del backend en Spring Boot	65
3.2.2.1 Comunicación con Ontop y Oracle	66

3.2.2.2 Metodología de diseño del prompt para la generación de consultas SPARQL	67
3.2.2 Implementación del frontend y visualización de resultados	69
3.2.3 Instalación en distintos entornos	70
3.3 Evaluación y validación	71
3.3.1 Metodología de evaluación	71
3.3.2 Resultados de la evaluación	72
3.3.3 Análisis de resultados	73
Parte 4	74
4. Desarrollo de la Detección de errores	74
4.1 Tipología de errores considerados	74
4.2 Representación semántica de errores en la ontología	75
4.3 Consultas SPARQL para detección de errores	76
4.4 Evaluación y validación	77
4.4.1 Metodología de evaluación	78
4.4.2 Resultados de la evaluación	79
4.4.3 Análisis de resultados	79
Parte 5	80
5. Desarrollo de Consultas personalizadas	80
5.1 Adaptación técnica específica para la generación de listados	80
5.2 Evaluación y validación	81
5.2.1 Metodología de evaluación	81
5.2.2 Resultados de la evaluación	82
5.2.3 Análisis de resultados	83
Parte 6	84
6. Conclusiones	84
6.1 Líneas de trabajo futuro	84
ANEXOS	85

Anexo I. Ontología “Atención al Ciudadano” (ontoAC)	85
Anexo II. Modelo base de datos AC	88
Anexo III. Doc. Backend y frontend “Botón Ayuda Inteligente”	91
Anexo IV. Archivo .odba y Ontop	102
Anexo V. Doc. Backend y frontend “Consultas personalizadas”	104
Anexo VI. Evaluación “Botón de Ayuda Inteligente”	107
Anexo VII. Evaluación “Detección de errores”	110
Anexo VIII. Evaluación “Consultas personalizadas”	111
Referencias	113

# Parte 1

## 1. Introducción

En la actualidad, la administración pública, independientemente de su ámbito o nivel, emplea aplicaciones web para gestionar y publicar información dirigida a la ciudadanía a través de *portales oficiales*<sup>1</sup>. Estas plataformas permiten a los usuarios acceder a diversos servicios y trámites administrativos, como solicitudes de empleo público, enlaces informativos, oposiciones, guías de funcionarios o procedimientos en áreas clave como educación, justicia y sanidad. Además, estas herramientas no solo facilitan la relación entre la administración y la ciudadanía, sino que también proporcionan funcionalidades esenciales para los propios empleados públicos, incluyendo interoperabilidad con sistemas gubernamentales como el Directorio Común DIR3 o el Sistema de Información Administrativa (SIA). Cabe señalar que, dado que la aplicación utilizada pertenece a un entorno real de producción de un cliente, no es posible mostrar capturas ni detalles específicos en este trabajo por motivos de confidencialidad.

A pesar de su utilidad, los sistemas actuales suelen estar desarrollados como aplicaciones web tradicionales, basadas en bases de datos relacionales y sin integración de inteligencia artificial. La ayuda disponible para el usuario en estos entornos es, por lo general, estática y limitada: consiste en botones de ayuda repartidos por distintas pantallas que, al ser activados, abren ventanas modales con fragmentos de contenido HTML predefinido. Este enfoque no solo implica una repetición innecesaria de la información entre secciones, sino que también impide ofrecer respuestas ajustadas al contexto específico de la acción que está realizando el usuario. Para superar estas limitaciones, se propone sustituir este modelo por un sistema de ayuda inteligente y dinámica, capaz de adaptarse al entorno y necesidades del usuario mediante técnicas de inteligencia artificial y conocimiento semántico.

Otro desafío recurrente en este tipo de sistemas es la detección y gestión de errores, especialmente cuando se introducen nuevas versiones, datos o configuraciones. En muchos casos, estos cambios pueden generar fallos funcionales o inconsistencias que afectan negativamente a la operatividad del sistema y a la calidad del servicio ofrecido.

---

<sup>1</sup> Portal de Atención a la Ciudadanía de la Generalitat Valenciana: [https://www.gva.es/es/web/atencio\\_ciudadania](https://www.gva.es/es/web/atencio_ciudadania)

Ante esta situación, se plantea la necesidad de incorporar un sistema que permita identificar de forma anticipada errores en la estructura lógica de los datos, mediante consultas semánticas que detecten condiciones anómalas o incumplimientos de reglas del dominio. Esta validación no solo mejora la calidad de los datos publicados, sino que permite actuar de forma proactiva antes de que los errores lleguen al usuario final.

Por último, otro aspecto que limita la experiencia del usuario es la dificultad para consultar y extraer información de manera autónoma dentro del sistema. Las opciones disponibles suelen requerir conocimientos técnicos o seguir rutas poco intuitivas, lo que restringe el acceso a los datos por parte de usuarios no especializados. Esta falta de accesibilidad obstaculiza la explotación eficiente de la información pública y ralentiza los procesos de consulta o generación de informes. Por ello, se plantea la necesidad de incorporar un sistema que facilite la interacción directa con los datos mediante mecanismos más naturales y adaptados al nivel del usuario, permitiendo obtener información personalizada de forma sencilla y rápida.

La Web Semántica, al estructurar los datos con significado explícito mediante ontologías, proporciona una base idónea para la integración de estos mecanismos. Permite enlazar conceptos, inferir conocimiento, validar la coherencia del sistema y mejorar la comunicación entre usuarios y *plataformas* [Zaveri et al., 2016; Kontokostas et al., 2014; Calvanese et al., 2017]. Aplicada al contexto administrativo, esta tecnología potencia la construcción de asistentes inteligentes, la automatización en la detección de errores y la personalización de consultas de información.

Este trabajo plantea el desarrollo de un modelo semántico escalable y adaptable, basado en tecnologías de Web Semántica e inteligencia artificial, orientado a mejorar la eficiencia y calidad de los sistemas de atención ciudadana. Aunque el enfoque propuesto puede aplicarse a distintos contextos sin depender de una aplicación específica, su evaluación se ha realizado mediante un conjunto de datos ficticios diseñados para simular escenarios reales dentro del ámbito del Sistema de Información de Atención a la Ciudadanía (SIAC)<sup>2</sup> de la Generalitat Valenciana (GVA). El objetivo es demostrar el potencial del modelo para mejorar la gestión de datos administrativos, la experiencia del usuario y la eficiencia en la toma de decisiones. En definitiva, dar inteligencia a la Web.

---

<sup>2</sup> Atención a la Ciudadanía de la Generalitat ValencianaGeneralitat Valenciana. (n.d.). *Sistema de Información de Atención a la Ciudadanía (SIAC)*. <https://rendicicomptes.gva.es/es/sistema-informacio-atencio-ciutadania>:

## 1.1 Motivación

La motivación de este trabajo nace de mi participación directa en el Sistema de Información de Atención a la Ciudadanía (SIAC), utilizado por la Generalitat Valenciana (GVA) para gestionar información administrativa antes de su publicación en el portal público. Desde mi rol funcional, he podido observar con detalle las limitaciones del sistema actual, tanto desde el punto de vista del usuario final como desde de los equipos técnicos que lo mantienen.

A lo largo del tiempo, he identificado carencias en aspectos clave como la falta de una ayuda contextualizada, la dificultad para detectar errores conceptuales tras cada publicación, o las restricciones en la consulta autónoma de datos. Esta experiencia diaria me llevó a plantearme una pregunta esencial:

Si la Web Semántica permite estructurar la información de manera inteligible para las máquinas, ¿por qué no aprovechar este enfoque para conceptualizar los datos y, a partir de ahí, aplicar técnicas de inteligencia artificial?

De esta reflexión surge la propuesta del presente Trabajo de Fin de Máster: explorar cómo la combinación de Web Semántica e IA podría aportar inteligencia real a sistemas administrativos ya existentes, optimizando su funcionamiento sin necesidad de partir de cero. No se trata solo de una hipótesis teórica, sino de una oportunidad concreta de mejora identificada en un entorno real, que podría extenderse a muchas otras aplicaciones similares en el ámbito público.

## 1.2 Objetivos

El objetivo de este trabajo es explorar cómo la combinación de Web Semántica e Inteligencia Artificial (IA) puede contribuir a mejorar la eficiencia y la calidad de los sistemas de atención al ciudadano utilizados en la administración pública. A partir de esta premisa, se plantean los siguientes objetivos específicos:

### Conceptualizar el dominio mediante Web Semántica

- Desarrollar una ontología del dominio que permita estructurar el conocimiento del sistema y facilitar la inferencia automática.
- Validar la ontología mediante el uso de razonadores semánticos (como Pellet o HermiT) y consultas SPARQL, garantizando su coherencia, consistencia y aplicabilidad práctica.

#### Aplicar técnicas de IA para dotar de inteligencia al sistema

- Integrar modelos de inteligencia artificial basados en Web Semántica para optimizar la gestión de información y facilitar la interacción con el usuario.

#### Diseñar e implementar un prototipo funcional con tres mejoras clave

- *Ayuda inteligente*: Sustituir la ayuda estática por un sistema contextualizado e interactivo que se adapte a la acción al usuario.
- *Detección de errores*: Identificar y reportar incoherencias conceptuales en los datos, como registros incompletos, formatos incorrectos o duplicidades, mediante una prueba de concepto funcional.
- *Consultas personalizadas*: Permitir que los usuarios realicen preguntas en lenguaje natural y obtengan información personalizada conectada al modelo de datos.

#### Evaluar el impacto del modelo propuesto

- Comparar la eficiencia y usabilidad del sistema antes y después de incorporar las mejoras.
- Definir métricas cuantitativas y cualitativas para valorar el impacto: tiempo medio de respuesta, precisión de la ayuda, tipología de errores para la detección de errores.

#### Identificar posibles limitaciones

- Detectar y documentar riesgos asociados a la implementación, como la carga computacional, posibles sesgos de los modelos o barreras en la adopción del sistema por parte de los usuarios.

### 1.3 Estado del arte

En esta sección se realiza una revisión de los enfoques y estudios previos sobre Web Semántica, ontologías y tecnologías de Inteligencia Artificial aplicadas a la gestión de información en entornos administrativos. El objetivo es analizar estas tecnologías para establecer una base teórica que justifique la viabilidad de la propuesta de este trabajo.

Se examinan las principales herramientas y estándares de la Web Semántica, que permiten estructurar y organizar la información de manera eficiente, mejorando su interoperabilidad y reutilización. Además, se aborda el uso de la IA para optimizar

procesos administrativos, facilitando la automatización de tareas y la generación de conocimiento a partir de datos estructurados.

Este análisis permitirá contextualizar las mejoras propuestas en este trabajo, estableciendo un marco teórico que sustente el desarrollo de una ayuda inteligente, la detección automática de errores y un asistente basado en lenguaje natural dentro del sistema de gestión de información administrativa.

### 1.3.1 Enfoques y estudios relacionados

En los últimos años, diversas investigaciones han explorado el uso de la Web Semántica e Inteligencia Artificial en distintos dominios, con el propósito de estructurar el conocimiento y mejorar la toma de decisiones en sistemas complejos. A continuación, se presentan estudios relevantes que sustentan la propuesta de este trabajo:

#### ***Interfaz conversacional para la recolección de datos en la Web Semántica***

*Firmenich (2023)* presenta una arquitectura para la integración de chatbots y sistemas de procesamiento de lenguaje natural (NLP) con la Web Semántica, con el objetivo de mejorar la interacción entre usuarios y bases de datos estructuradas mediante consultas conversacionales. Su enfoque permite traducir preguntas en lenguaje natural a consultas sobre ontologías RDF/OWL, facilitando la recuperación automatizada de información.

#### ***Distilling ontologies from large languages model***

*Huaranga Junco, E. J. (n.d.)* exploran cómo los LLMs como GPT-4 pueden utilizarse para extraer y refinar ontologías desde fuentes no estructuradas. Este enfoque apoya la construcción automática del modelo ontológico propuesto en este trabajo, reduciendo la intervención manual y mejorando la escalabilidad.

#### ***Diseño de un modelo explicativo basado en ontologías aplicado a un chatbot conversacional***

*Arteaga Meléndez (2023)* desarrolla una arquitectura de chatbot basada en ontologías, que permite generar respuestas explicativas y contextualizadas. El modelo aprovecha la Web Semántica para estructurar el conocimiento del dominio y proporcionar interacciones más precisas y adaptadas al contexto.

#### ***Creación de un prototipo de chatbot que permita interactuar con la historia del Ecuador registrada en periódicos antiguos***

*Ciancio (2024)* diseña un chatbot conversacional basado en NLP, enfocado en recuperar información histórica de periódicos digitalizados. El sistema permite consultar datos mediante lenguaje natural, optimizando la búsqueda y accesibilidad.

### **Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional**

*Celi-Párraga, Varela-Tapia, Acosta-Guzmán y Montaño-Pulzara (2021)* analizan las principales técnicas de NLP aplicadas a chatbots y asistentes virtuales, incluyendo el reconocimiento de entidades (NER), transformers como BERT y GPT, y generación de respuestas contextuales.

### **Sistema de ayuda a la decisión basado en ontologías para el diagnóstico agrícola**

*Lagos Ortiz, K. A. (2020)* propone un sistema de ayuda a la decisión para la prevención de enfermedades en cultivos, basado en ontologías que modelan síntomas, tratamientos y relaciones entre conceptos. Este enfoque demuestra cómo las ontologías pueden emplearse para generar recomendaciones personalizadas.

### **Explotación de la semántica de dominio orientada a la integración, estandarización y análisis de datos**

*Paneque Romero (2022)* presenta un marco basado en Web Semántica que facilita la integración, estandarización y análisis de datos mediante ontologías, promoviendo la interoperabilidad entre sistemas. Esta visión resulta útil para desarrollar consultas en lenguaje natural sobre datos administrativos estructurados.

### **MethoOntoChat: Asistente conversacional del proceso metodológico de creación de ontologías basado en modelos del lenguaje**

*Gómez de Agüero Muñoz (2024)* presenta un asistente conversacional para apoyar la creación de ontologías, utilizando tecnologías como RDF, OWL y SPARQL. Su enfoque demuestra cómo la Web Semántica puede emplearse para estructurar y gestionar información en entornos administrativos complejos.

### **Towards next-generation urban decision support systems through AI-powered construction of scientific ontology using large language models—A case in optimizing intermodal freight transportation.**

Uno de los enfoques recientes en la generación y estructuración del conocimiento mediante IA es el propuesto por *Tupayachi et al. (2024)*. En su estudio, se plantea una metodología innovadora para la construcción de ontologías científicas

utilizando Modelos de Lenguaje de Gran Escala (LLMs), como GPT-4. El trabajo se centra en mejorar los sistemas de soporte a la toma de decisiones en entornos urbanos, aplicando IA para extraer, estructurar y organizar el conocimiento a partir de grandes volúmenes de datos textuales. Los principales aportes incluyen la automatización del modelado ontológico con LLMs, la integración de Web Semántica e IA para mejorar la interoperabilidad, y la validación de la ontología mediante inferencia semántica y consultas estructuradas.

***Revisión de los chatbots basados en inteligencia artificial en la administración pública: Hacia una arquitectura para el gobierno***

*Martínez (2022)* realiza una revisión sistemática sobre el uso de chatbots en la administración pública y propone un marco arquitectónico basado en NLP, Web Semántica y aprendizaje automático.

***Sistema de generación de consultas SQL basado en peticiones de lenguaje natural a partir de un chat-bot***

*Brito Karadjian, B. J. (2024)* propone el desarrollo de un sistema que permite a los usuarios formular consultas en lenguaje natural, las cuales son interpretadas y convertidas en consultas SQL mediante un chatbot. El sistema integra herramientas como Streamlit para la interfaz de usuario, modelos de lenguaje de OpenAI, LangChain para la gestión de consultas SQL y una base de datos PostgreSQL. El objetivo es mejorar la eficiencia y experiencia del usuario al interactuar con bases de datos sin necesidad de conocimientos técnicos en SQL.

# Parte 2

## 2. Modelo Semántico y Arquitectura del Sistema basado en Ontología

Se especifica el modelo semántico desarrollado como base del sistema y su integración dentro de una arquitectura que permite consultar datos reales mediante tecnología ontológica. En primer lugar, se detalla el proceso completo de construcción de la ontología “Atención al Ciudadano”. A continuación, se describe la arquitectura técnica que permite aplicar dicha ontología sobre una base de datos relacional real, mediante un motor de consulta semántica (Ontop) y un backend modular basado en Spring Boot.

Cabe señalar que tanto la ontología como el modelo de datos implementado constituyen prototipos funcionales diseñados con fines académicos y demostrativos. Su complejidad ha sido acotada deliberadamente para poder desarrollar, probar y validar el enfoque en el marco del presente Trabajo de Fin de Máster. No obstante, la solución propuesta está concebida para ser escalable y adaptable a sistemas reales de mayor envergadura, en caso de que se decidiera su evolución hacia un entorno productivo.

### 2.1 Creación de la Ontología “Atención al Ciudadano” (ontoAC)

Las siguientes secciones documentan la definición detallada de la ontología utilizada en el sistema, enfocándose en la representación de las entidades: Personas, Departamentos, CriteriosBusqueda, Operación, Registros, TramiteServicio y sus fases, EmpleoPublico con sus etapas, EntidadLocales, AyudaCampo.

La ontología permite estructurar y gestionar de manera eficiente la información relacionada con:

- *Personas*, que trabajan en la administración pública de la GVA, y los departamentos a los que están asignadas.
- *Departamentos*, en los que se organiza la estructura administrativa de la GVA, con sus funciones, normativa de creación, ubicación y contacto.
- *Registros*, donde los ciudadanos pueden presentar documentación dirigida a la administración.
- *Trámites y servicios*, que la Generalitat ofrece a los ciudadanos, organizados por fases y procedimientos específicos.

- *Empleo público*, orientado a proporcionar información sobre ofertas o convocatorias de empleo público a nivel de la Comunidad Valenciana.
- *Entidades locales*, incluyendo ayuntamientos, mancomunidades, entidades inframunicipales, diputaciones y núcleos de población de la Comunidad Valenciana.
- *Criterios de búsqueda*, definidos para permitir búsquedas avanzadas de información dentro de cada módulo o entidad.
- *Operaciones*, que representa las acciones que se pueden realizar sobre cada módulo o entidad, tales como consultar, añadir, modificar, eliminar e imprimir información.
- *AyudaCampo*, representa la información contextual y funcional asociada a cada campo del sistema, con el objetivo de proporcionar orientación inteligente al usuario durante la interacción con los distintos módulos administrativos. Esta entidad no forma parte del dominio administrativo en sí, sino que actúa como soporte transversal, ya que no está vinculada exclusivamente a un único módulo o entidad, sino que da apoyo a todos ellos para la generación de ayudas explicativas dentro del sistema.

Para su desarrollo, se utiliza la metodología NeOn (Suárez-Figueroa & Gómez-Pérez, 2012), la cual ofrece un conjunto de tareas específicas para la creación y mantenimiento de ontologías. Sin embargo, su aplicación se limita a las fases consideradas necesarias para este trabajo, dado que la ontología se utiliza posteriormente para realizar un mapeo con una base de datos relacional. Este mapeo permite extraer la información almacenada siguiendo la estructura conceptual establecida por la ontología.

### **2.1.1 Especificación de requisitos**

Se definen los requisitos funcionales y no funcionales que la ontología debe cumplir, así como sus usos previstos, usos finales y preguntas de competencia. Este proceso tiene como objetivo establecer una base sólida para la creación de la ontología y garantizar su alineación con los objetivos planteados.

El desarrollo de esta especificación sigue las pautas establecidas por el documento *Ontology Requirements Specification Document (ORSD)*, propuesto en la metodología NeOn.

La recolección y análisis de requisitos se realiza mediante un proceso de revisión interna y análisis conceptual basado en la información pública existente sobre el

sistema de atención al ciudadano en el portal de la GVA (Generalitat Valenciana). Este análisis se lleva a cabo considerando las funcionalidades actuales del sistema, así como las necesidades detectadas para proporcionar un modelo ontológico adecuado que estructure correctamente la información relevante.

*Actividad 1: Identificación del propósito, alcance lenguaje implementación de la ontología.*

### Propósito

El propósito principal de la ontología es estructurar y contextualizar la información relacionada con el sistema de atención al ciudadano de la Comunidad Valenciana de manera que sea inteligible por sistemas computacionales. Esto permite su uso para:

- Facilitar la obtención de información inteligente a través de un asistente inteligente capaz de responder consultas formuladas en lenguaje natural.
- Proveer ayuda dinámica que sustituya y mejore la actual ayuda estática del sistema.
- Generar listados personalizados y detallados a partir de la información registrada en el sistema.
- Facilitar la implementación de técnicas de Inteligencia Artificial, como la detección automática de errores en la plataforma.

Esta ontología sirve como base para la interacción eficiente con el sistema de atención al ciudadano y la expansión futura de sus capacidades.

### Alcance

El alcance de la ontología se limita, en esta primera etapa, a un subconjunto de la información relevante del sistema de atención al ciudadano, manteniendo la posibilidad de ampliación y actualización futura.

Las áreas específicas que abarca esta ontología son:

- Personas y sus relaciones con departamentos.
- Departamentos y órganos relacionados.
- Registros y entidades locales disponibles.
- Trámites y servicios en los que las personas pueden iniciar, gestionar o consultar procedimientos.

- Empleo público al que los ciudadanos pueden optar mediante convocatorias y procedimientos específicos.
- Criterios de búsqueda que facilitan la obtención de información relevante sobre cada uno de los módulos.
- Operaciones permitidas sobre cada módulo o entidad.
- Ayuda para cada uno de los campos del sistema.

La ontología se desarrollada en formato OWL (Web Ontology Language) y RDF (Resource Description Framework) (W3C, 2012; W3C, 2014), facilitando la interoperabilidad y el uso en sistemas basados en Web Semántica.

#### *Actividad 2: Identificación de los usuarios finales previstos*

La ontología está diseñada principalmente para su uso en un sistema de gestión de la información que posteriormente será visualizada por el ciudadano a través del portal de la GVA (Generalitat Valenciana). Sin embargo, también puede ser utilizada directamente desde el portal ciudadano, ya que ambos sistemas comparten la misma información, pero desde contextos diferentes: uno orientado a la gestión interna y otro a la consulta pública.

De acuerdo con lo anterior, se identifican tres tipos principales de usuarios:

- Usuario 1: Profesionales que buscan información relacionada con la gestión administrativa (Uso Interno).

Este grupo incluye a usuarios que necesitan consultar información sobre campos y procedimientos específicos para poder realizar su trabajo adecuadamente. Cubre las siguientes necesidades:

- Consultar documentación detallada sobre procedimientos, servicios, registros y empleo público.
- Obtener información sobre cómo completar o modificar campos específicos en formularios o interfaces.
- Verificar la visibilidad y validez de la información.
- Acceder a la ayuda inteligente para conocer cómo utilizar ciertos procedimientos o consultas.
- Acceder al asistente inteligente para obtener listados personalizados.
- Detección de errores en el sistema.

Ejemplos de usuarios:

- Administrador Técnico: Consulta información para implementar cambios técnicos o corregir errores en el sistema.
  - Administrador Funcional: Accede a la información para asegurar que los procedimientos se publican correctamente y son accesibles al ciudadano.
  - Documentalista: Verifica y consulta datos estructurados para garantizar la consistencia de la información en diferentes secciones.
  - Consultor: Consulta la ontología para comprobar cómo se define la información que se presenta en el sistema.
- Usuario 2: Operadores que proporcionan información al ciudadano (Uso Intermedio).

Este grupo incluye a profesionales que utilizan la ontología para responder a consultas realizadas por ciudadanos, pero no interactúan directamente con el sistema. Obtienen respuestas rápidas a preguntas específicas formuladas por ciudadanos.

- Usuario 3: Ciudadanos que consultan información publicada en el portal GVA (Uso Público)

Ciudadanos que acceden al portal de la GVA para realizar consultas o trámites relacionados con la atención al ciudadano. Realizan consultas en lenguaje natural a través de un asistente inteligente o formularios de consulta.

#### *Actividad 3: Identificación de los usos previstos*

El uso previsto de la ontología es proporcionar un marco semántico que permita la creación de inteligencia en la Web mediante el uso de tecnologías de Web Semántica e Inteligencia Artificial. La ontología tiene como propósito estructurar, contextualizar y relacionar la información proveniente de la base de datos relacional Oracle, para hacerla accesible y comprensible por sistemas inteligentes que puedan operar sobre ella.

La ontología debe ser capaz de representar el modelo de información del sistema de atención al ciudadano, sirviendo como un puente entre la base de datos relacional y las aplicaciones inteligentes que se desarrollen sobre ella: ayuda dinámica e inteligente que sustituya a la ayuda estática, detección de errores en el sistema a partir de la información ontológica y generación de listados personalizado en formato como Excel o CSV, de datos en base a consultas realizadas sobre la ontología.

#### *Actividad 4: Identificación de los requisitos*

En esta tarea se identifican tanto los requisitos funcionales como los requisitos no funcionales que deben guiar la creación de la ontología. Los requisitos funcionales se definen en forma de preguntas de competencia, que establecen la base del conocimiento que la ontología debe representar. Por otro lado, los requisitos no funcionales describen aspectos técnicos y metodológicos relacionados con la construcción, implementación y mantenimiento de la ontología.

#### Requisitos no funcionales:

Las características técnicas y metodológicas que debe cumplir la ontología para asegurar su calidad, coherencia y escalabilidad son:

##### 1. Idioma y formato de la ontología:

- La ontología se desarrolla en castellano, utilizando OWL (Web Ontology Language) y representada en formato RDF (Resource Description Framework).

##### 2. Escalabilidad:

- La ontología debe ser extensible para permitir la incorporación de nuevas clases, propiedades y relaciones sin comprometer su coherencia.
- Debe garantizar que la adición de nuevas áreas de conocimiento o módulos sea posible sin afectar negativamente el rendimiento del sistema.

##### 3. Compatibilidad e interoperabilidad:

- La ontología debe ser compatible con tecnologías de Web Semántica, permitiendo consultas SPARQL y su integración con sistemas externos mediante mapeo ontológico con la base de datos Oracle.

##### 4. Rendimiento:

- Las consultas realizadas sobre la ontología deben procesarse de manera eficiente, incluso con grandes volúmenes de datos.
- Debe garantizarse un rendimiento adecuado en escenarios de uso intensivo por múltiples usuarios.

## 5. Documentación y trazabilidad:

- El proceso de construcción y actualización de la ontología debe estar claramente documentado mediante el documento Ontology Requirements Specification Document (ORSD).
- Toda modificación debe registrarse para asegurar su trazabilidad.

### Requisitos funcionales:

Preguntas de competencia que establecen el conocimiento que la ontología debe ser capaz de representar y responder. Estas preguntas están directamente relacionadas con la información que expresa cada propiedad de la ontología, de acuerdo con cada uno de sus subdominios. Se recogen en la Tabla 1, Tabla 2, Tabla 3, Tabla 4, Tabla 5, Tabla 6 y Tabla 7.

Tabla 1. Preguntas de Competencia sobre *Personas*.

Identificador	Pregunta de competencia	Posibles resultados
PER_1	¿Nombre y apellidos persona X?	Ana María, García Sánchez
PER_2	¿Cuál es el correo de la persona X?	anapg@gva.es
PER_3	¿Qué tipo persona es la persona X?	Interno
PER_4	¿Qué visibilidad tiene la persona X?	Pública
PER_5	¿En qué unidad trabaja la persona X?	Departamento Jurídico
PER_6	¿Dónde está ubicada la persona X?	Calle Mayor 1
PER_7	¿CP ubicación persona X?	46019
PER_8	¿Municipio, provincia persona X?	Valencia, Valencia
PER_9	¿Planta ubicación persona X?	Planta 1
PER_10	¿Despacho ubicación persona X?	Despacho 101
PER_11	¿Teléfono externo persona X?	987654321
PER_12	¿Teléfono interno persona X?	1234
PER_13	¿Fax externo persona X?	912345678
PER_14	¿Fax interno persona X?	5678
PER_15	¿Identificación persona X?	11111111H
PER_16	¿Nacionalidad persona X?	Española
PER_17	¿Género persona X?	Femenino
PER_18	¿Es un alto cargo persona X?	No
PER_19	¿Cargo persona X?	Asesora
PER_20	¿La persona X está adscrita a un departamento?	Sí
PER_21	¿Función de la persona X?	Asesoría Jurídica
PER_22	¿Código interno de la persona X?	1
PER_23	¿Código departamento del que está adscrito persona X?	101
PER_24	¿Nombre de personas adscritas al departamento X?	Ana María, Javier, Raúl
PER_25	¿Conselleria persona X?	Conselleria de Justicia

Tabla 2. Preguntas de Competencia sobre *Departamentos*.

Identificador	Pregunta de competencia	Posibles resultados
DEP_1	¿Código y nombre departamento X?	101, Departamento Jurídico
DEP_2	¿Cuál es el correo departamento X?	juridico@gva.es
DEP_3	¿Fecha creación departamento X?	28/03/25
DEP_4	¿Reglamento creación departamento X?	Decreto 123/2025
DEP_5	¿Horario departamento X?	Lunes a Viernes 8:00 - 15:00
DEP_6	¿Dirección departamento X?	Edificio A, Planta 1
DEP_7	¿CP ubicación departamento X?	46001
DEP_8	¿Municipio departamento X?	Valencia
DEP_9	¿Provincia departamento X?	Valencia
DEP_10	¿Planta ubicación departamento X?	13
DEP_11	¿Teléfono externo departamento X?	987654321
DEP_12	¿Teléfono interno departamento X?	1234
DEP_13	¿Fax externo departamento X?	912345678
DEP_14	¿Fax interno departamento X?	5678
DEP_15	¿Tipo centro departamento X?	Jurídico
DEP_16	¿Código interno departamento X?	27014
DEP_17	¿Rango departamento X?	Dirección General
DEP_18	¿Conselleria departamento X?	Conselleria de Justicia

Tabla 3. Preguntas de Competencia sobre *CriterioBusqueda*.

Identificador	Pregunta de competencia	Posibles resultados
CRB_1	¿Nombre criterio de búsqueda X?	Nombre Parcial
CRB_2	¿Descripción criterio de búsqueda X?	Permite buscar por nombres parcialmente coincidentes.
CRB_3	¿Tipo de comodín criterio de búsqueda X?	VALOR%
CRB_4	¿Uso comodín X para el criterio de búsqueda X?	Ejemplo: Ana% devuelve Ana, Ana María, Ana Isabel.
CRB_5	¿Campo que conforman el criterio de búsqueda X?	PER_NOMBRE

Tabla 4. Preguntas de Competencia sobre *Operacion*.

Identificador	Pregunta de competencia	Posibles resultados
OPE_1	¿Nombre operación X?	Consultar
OPE_2	¿Descripción operación X?	Permite visualizar información registrada en el sistema.
OPE_3	¿Operaciones existentes?	Consultar, Insertar, Modificar, Eliminar, Exportar

Tabla 5. Preguntas de Competencia sobre *Registros*.

Identificador	Pregunta de competencia	Posibles resultados
REG_1	¿Nombre del registro X?	Registro General
REG_2	¿Cuál es el correo registro X?	registro@gva.es
REG_3	¿Horario registro X?	Lunes a viernes 9:00 - 14:00
REG_4	¿Dónde está ubicado registro X?	Calle del Sol 5
REG_5	¿CP ubicación registro X?	12003
REG_6	¿Municipio registro X?	Castellón
REG_7	¿Teléfono externo registro X?	912345679
REG_8	¿Teléfono interno registro X?	6789
REG_9	¿Fax externo registro X?	987654322
REG_10	¿Fax interno registro X?	5678
REG_11	¿Tipo registro X?	Externo
REG_12	¿Código interno registro X?	152
REG_13	¿Conselleria del registro X?	Conselleria de Gobernación
REG_14	¿Provincia registro X?	Castellón
REG_15	¿Código departamento al que está asignado el registro X?	101

Tabla 6. Preguntas de Competencia sobre *TramiteServicio*.

Identificador	Pregunta de competencia	Posibles resultados
TRA_1	¿Código del trámite X?	1
TRA_2	¿Departamento asignado al trámite X?	Departamento Jurídico
TRA_3	¿Descripción del trámite X?	Solicitud de asesoramiento jurídico para procedimientos administrativos complejo
TRA_4	¿Destinatario del trámite X?	Ciudadanía
TRA_5	¿Objetivo del trámite X?	Proporcionar asistencia jurídica y orientación en trámites legales.
TRA_6	¿Es un trámite o un servicio?	Trámite
TRA_7	¿Código del servicio X?	4
TRA_8	¿Código del trámite tipo servicio asignado al servicio X?	4
TRA_9	¿Descripción del servicio X?	Servicio de atención legal
TRA_10	¿Destinatario del servicio X?	Ciudadanía
TRA_11	¿Objetivo del servicio X?	Registro de documentación oficial
TRA_12	¿Enlace del servicio X?	<a href="http://gva.es/servicio-info1">http://gva.es/servicio-info1</a>
TRA_13	¿Fecha fin del servicio X?	07/04/25
TRA_14	¿Fecha inicio del servicio X?	17/04/25
TRA_15	¿Información complementaria del servicio X?	Puede requerirse documentación adicional según el caso.
TRA_16	¿Interesados en servicio X?	Funcionariado
TRA_17	¿Normativa en servicio X?	Decreto 12/2023 del Consell
TRA_18	¿Objeto del servicio X?	Asistencia legal gratuita
TRA_19	¿Plazo del servicio X?	15 días hábiles
TRA_20	¿Requisitos del servicio X?	Presentar copia del DNI y formulario oficial

TRA_21	¿Normativa de la fase inicio del trámite X?	Ley 12/2025 de Procedimientos Administrativos.
TRA_22	¿Objeto de la fase inicio del trámite X?	Solicitud de asesoramiento jurídico.
TRA_23	¿Plazo presentación de la fase inicio del trámite X?	Del 1 al 31 de cada mes.
TRA_24	¿Requisitos de la fase inicio del trámite X?	Presentar formulario de solicitud y documentación pertinente.
TRA_25	¿Tasa pago de la fase inicio del trámite X?	50 EUR
TRA_26	¿Interesados en la fase inicio del trámite X?	Ciudadanía y Empresas
TRA_27	¿Forma de presentación de la fase inicio del trámite X?	Telemática y Presencial
TRA_28	¿Fecha inicio de la fase inicio del trámite X?	01/03/25
TRA_29	¿Fecha fin de la fase inicio del trámite X?	31/03/25
TRA_30	¿Enlace de la fase inicio del trámite X?	<a href="http://example.org/asesoramiento-juridico">http://example.org/asesoramiento-juridico</a>
TRA_31	¿Criterios de valoración de la fase de instrucción del trámite X?	Baremo de criterios objetivos publicados previamente.
TRA_32	¿Departamento de tramitación de la fase de instrucción del trámite X?	Departamento Técnico
TRA_33	¿Enlace de la fase de instrucción del trámite X?	<a href="http://example.org/instrucion-tramite2">http://example.org/instrucion-tramite2</a>
TRA_34	¿Información de la fase de instrucción del trámite X?	La tramitación se realiza de forma telemática con revisión documental.
TRA_35	¿Normativa de la fase de instrucción del trámite X?	Normativa específica del procedimiento evaluado.
TRA_36	¿Enlace de la subfase de alegación del trámite X?	<a href="http://example.org/alegacion-tramite1">http://example.org/alegacion-tramite1</a>
TRA_37	¿Fecha inicio de la subfase de alegación del trámite X?	07/04/25
TRA_38	¿Fecha fin de la subfase de alegación del trámite X?	14/04/25
TRA_39	¿Forma de presentación de la subfase de alegación del trámite X?	Telemática
TRA_40	¿Normativa de la subfase de alegación del trámite X?	Ley 39/2015 del Procedimiento Administrativo Común.
TRA_41	¿Objeto de la subfase de alegación de la fase instrucción X?	Ofrecer plazo para subsanar errores o aportar información adicional.
TRA_42	¿Plazo de la subfase de alegación del trámite X?	10 días hábiles
TRA_43	¿Departamento resolución de la fase de finalización del trámite X?	Unidad de Resolución de Conflictos
TRA_44	¿Descripción de la fase de finalización del trámite X?	Finalización del procedimiento con resolución administrativa.
TRA_45	¿Enlace de la fase de finalización del trámite X?	<a href="http://example.org/finalizacion-2">http://example.org/finalizacion-2</a>
TRA_46	¿Plazo máximo de la fase de finalización del trámite X?	45 días hábiles
TRA_47	¿Procedimiento de cobro de la fase de finalización del trámite X?	Mediante domiciliación bancaria autorizada.

TRA_48	¿Recurso de la fase de finalización del trámite X?	Recurso de reposición en el plazo de 1 mes.
TRA_49	¿Sanciones de la fase de finalización del trámite X?	Ninguna sanción prevista.
TRA_50	¿Cuantía de la fase de finalización del trámite X?	25 EUR
TRA_51	¿Enlace de la subfase de justificación y cobro de la fase fin X?	<a href="http://example.org/justificacion-3">http://example.org/justificacion-3</a>
TRA_52	¿Fecha inicio de la subfase de justificación y cobro del trámite X?	09/06/25
TRA_53	¿Fecha fin de la subfase de justificación y cobro del trámite X?	18/06/25
TRA_54	¿Forma de presentación de la subfase de justificación y cobro del trámite X?	Ambas modalidades
TRA_55	¿Normativa de la subfase de justificación y cobro de la fase fin 4X?	Ley 9/2024
TRA_56	¿Objeto de la subfase de justificación y cobro del trámite X?	Verificar las condiciones que permiten el abono correspondiente
TRA_57	¿Plazo de la subfase de justificación y cobro del trámite X?	20 días hábiles
TRA_58	¿Normativa de justificación y cobro del trámite X?	Normativa de justificación administrativa del Consell

Tabla 7. Preguntas de Competencia sobre *EmpleoPublico*.

Identificador	Pregunta de competencia	Posibles resultados
EP_1	¿Código del empleo público X?	1
EP_2	¿Convocatoria empleo público X?	Convocatoria 2025 - Asesor Jurídico
EP_3	¿Departamento empleo público X?	Departamento Jurídico
EP_4	¿Descripción empleo público X?	Convocatoria para asesoría jurídica en la Conselleria
EP_5	¿Plaza empleo público X?	Asesoría jurídica en materias administrativas
EP_6	¿Número plaza empleo público X?	5
EP_7	¿Fecha creación empleo público X?	2025-03-30
EP_8	¿Requisito empleo público X?	Licenciatura en Derecho
EP_9	¿Prueba empleo público X?	Prueba teórica y práctica
EP_10	¿Titulación empleo público X?	Derecho
EP_11	¿Titulación específica empleo público X?	Derecho Administrativo
EP_12	¿Grupo empleo público X?	Grupo A1
EP_13	¿Tipo personal oferta empleo público X?	Funcionario
EP_14	¿Estado plazo presentación empleo público X?	Abierto
EP_15	¿Importe tasa empleo público X?	50€
EP_16	¿Reducción tasa empleo público X?	25%
EP_17	¿Método pago empleo público X?	Transferencia bancaria

EP_18	¿Información presentación empleo público X?	Presentación vía telemática
EP_19	¿Enlace informativo empleo público X?	<a href="http://gva.es/empleo-juridico">http://gva.es/empleo-juridico</a>
EP_20	¿Departamento al que pertenece el empleo público X?	101
EP_21	¿Fases del empleo público X?	Fase1, Fase2
EP_22	¿Nombre de la fase1 del empleo público X?	Presentación de Solicitudes
EP_23	¿Plazo abre de la fase1 del empleo público X?	2025-04-01
EP_24	¿Plazo cierre de la fase1 del empleo público X?	2025-04-30
EP_25	¿Fase actual la fase1 del empleo público X?	Sí
EP_26	¿Documentación de entrega de la fase1 del empleo público X?	Formulario de inscripción
EP_27	¿Fecha publicación de la fase1 del empleo público X?	2025-03-31
EP_28	¿Medio publicación de la fase1 del empleo público X?	Web Conselleria
EP_29	¿Enlace de la fase1 del empleo público X?	<a href="http://gva.es/solicitud-juridica">http://gva.es/solicitud-juridica</a>
EP_30	¿Etapa de la fase1 del empleo público X?	Primera Etapa

Tabla 8. Preguntas de Competencia sobre *EntidadLocales*.

Identificador	Pregunta de competencia	Posibles resultados
EEL_1	¿Tipo entidad local X?	Mancomunidad
EEL_2	¿Nombre entidad local X?	Mancomunidad de La Ribera
EEL_3	¿Descripción entidad local X?	Entidad supramunicipal que gestiona servicios mancomunados.
EEL_4	¿Dirección entidad local X?	Calle de las Mancomunidades 23
EEL_5	¿CP entidad local X?	6200
EEL_6	¿Municipio entidad local X?	Alzira
EEL_7	¿Provincia entidad local X?	Valencia
EEL_8	¿Comarca entidad local X?	Ribera Alta
EEL_9	¿Teléfono entidad local X?	+34 962408000
EEL_10	¿Fax entidad local X?	+34962408001
EEL_11	¿Correo entidad local X?	<a href="mailto:info@manriberalta.es">info@manriberalta.es</a>
EEL_12	¿Web entidad local X?	<a href="http://manriberalta.es">http://manriberalta.es</a>

Tabla 9. Preguntas de Competencia sobre *AyudaCampo*.

Identificador	Pregunta de competencia	Posibles resultados
AYU_1	¿Campo X?	PER NOMBRE
AYU_2	¿Nombre del campo X?	Nombre de la persona
AYU_3	¿Descripción del campo X?	Introduzca el nombre completo de la persona tal y como aparece en su documento de identidad. No se permiten apodos ni abreviaturas.
AYU_4	¿Ejemplo uso del campo X?	Ejemplo: Ana María Pérez
AYU_5	¿Módulo del campo X?	Personas
AYU_6	¿Es obligatorio el campo X?	Sí

AYU_7	¿Validación del campo X?	Debe contener solo letras y espacios. Máximo 50 caracteres.
-------	--------------------------	---

### *Tarea 5: Agrupación de los requisitos funcionales*

Los requisitos funcionales se agrupan de acuerdo con los subdominios declarados:

Subdominio	Tabla	Agrupación CMP
Personas	Tabla 1	Preguntas de competencia sobre los datos de las personas que trabajan en la administración pública de la GVA.
Departamentos	Tabla 2	Preguntas de competencia sobre los departamentos en la administración pública de la GVA.
CriterioBusqueda	Tabla 3	Preguntas de competencia sobre criterios de búsqueda de cada uno de los subdominios.
Operacion	Tabla 4	Operaciones que se aplican a cada uno de los subdominios.
Registros	Tabla 5	Preguntas de competencia sobre los registros existentes en la administración pública de la GVA.
TramiteServicio	Tabla 6	Preguntas de competencia sobre trámites y servicios dados de alta en la administración pública de la GVA.
EmpleoPublico	Tabla 7	Preguntas de competencia sobre empleo público (bolsas de trabajo, ofertas de empleo, oposiciones) dados de alta en la administración pública de la GVA.
EntidadLocales	Tabla 8	Preguntas de competencia sobre entidades locales perteneciente a la administración pública de la GVA.
AyudaCampo	Tabla 9	Preguntas de competencia sobre la ayuda de campos del sistema.

### *Actividad 6: Validación del conjunto de requisitos*

Se verifica que los requisitos sean coherentes, completos, consistentes y adecuados al propósito de la ontología. Se realiza un análisis para asegurar que cada pregunta de competencia es relevante y responde a una necesidad funcional clara.

### *Actividad 7: Priorización del conjunto de requisitos*

La prioridad de los requisitos funcionales ha sido el de las entidades de *Personas*, *Departamentos*, *CriterioBusqueda* y *Operación*. Estos se han tomado como base de la información, ya que a partir de dicha información se va ampliando la ontología con las siguientes entidades: *Registros*, *TramiteServicio*, *EmpleoPublico*, *EntidadLocales*, *AyudaCampo*.

## ***ORSD completo***

A continuación, se detalla al completo la plantilla de requisitos, expuesta en la Tabla 10.

Tabla 10. Plantilla ORSD completada.

Documento de Especificación de Requisitos de la Ontología	
1	<b>Propósito</b>
<p>El propósito principal de la ontología es estructurar y contextualizar la información relacionada con el sistema de atención al ciudadano de la Comunidad Valenciana de manera que sea inteligible por sistemas computacionales. Esto permite su uso para:</p> <ul style="list-style-type: none"><li>• Facilitar la obtención de información inteligente a través de un asistente inteligente capaz de responder consultas formuladas en lenguaje natural.</li><li>• Proveer ayuda dinámica que sustituya y mejore la actual ayuda estática del sistema.</li><li>• Generar listados personalizados y detallados a partir de la información registrada en el sistema.</li><li>• Facilitar la implementación de técnicas de Inteligencia Artificial, como la detección automática de errores en la plataforma.</li></ul> <p>Esta ontología sirve como base para la interacción eficiente con el sistema de atención al ciudadano y la expansión futura de sus capacidades.</p>	
2	<b>Alcance</b>
<p>El alcance de la ontología se limita, en esta primera etapa, a un subconjunto de la información relevante del sistema de atención al ciudadano, manteniendo la posibilidad de ampliación y actualización futura.</p> <p>Las áreas específicas que abarca esta ontología son:</p> <ul style="list-style-type: none"><li>• Personas y sus relaciones con departamentos.</li><li>• Departamentos y órganos relacionados.</li><li>• Registros y entidades locales disponibles.</li><li>• Trámites y servicios en los que las personas pueden iniciar, gestionar o consultar procedimientos.</li></ul>	

- Empleo público al que los ciudadanos pueden optar mediante convocatorias y procedimientos específicos.
- Criterios de búsqueda que facilitan la obtención de información relevante sobre cada uno de los módulos.
- Operaciones permitidas sobre cada módulo o entidad.
- Ayuda para cada uno de los campos del sistema.

### **3 Lenguaje de implementación**

La ontología se desarrollada en formato OWL (Web Ontology Language) y RDF (Resource Description Framework), facilitando la interoperabilidad y el uso en sistemas basados en Web Semántica.

### **4 Usuarios finales previstos**

La ontología está diseñada principalmente para su uso en un sistema de gestión de la información que posteriormente será visualizada por el ciudadano a través del portal de la GVA (Generalitat Valenciana). Sin embargo, también puede ser utilizada directamente desde el portal ciudadano, ya que ambos sistemas comparten la misma información, pero desde contextos diferentes: uno orientado a la gestión interna y otro a la consulta pública.

De acuerdo con lo anterior, se identifican tres tipos principales de usuarios:

- Usuario 1: Profesionales que buscan información relacionada con la gestión administrativa (Uso Interno).

Este grupo incluye a usuarios que necesitan consultar información sobre campos y procedimientos específicos para poder realizar su trabajo adecuadamente. Cubre las siguientes necesidades:

- Consultar documentación detallada sobre procedimientos, servicios, registros y empleo público.
- Obtener información sobre cómo completar o modificar campos específicos en formularios o interfaces.
- Verificar la visibilidad y validez de la información.
- Acceder a la ayuda inteligente para conocer cómo utilizar ciertos procedimientos o consultas.
- Acceder al asistente inteligente para obtener listados personalizados.
- Detección de errores en el sistema.
- Ejemplos de usuarios:
  - Administrador Técnico: Consulta información para implementar cambios técnicos o corregir errores en el sistema.
  - Administrador Funcional: Accede a la información para asegurar que los procedimientos se publican correctamente y son accesibles al ciudadano.
  - Documentalista: Verifica y consulta datos estructurados para garantizar la consistencia de la información en diferentes secciones.
  - Consultor: Consulta la ontología para comprobar cómo se define la información que se presenta en el sistema.

- Usuario 2: Operadores que proporcionan información al ciudadano (Uso Intermedio).

<p>Este grupo incluye a profesionales que utilizan la ontología para responder a consultas realizadas por ciudadanos, pero no interactúan directamente con el sistema. Obtienen respuestas rápidas a preguntas específicas formuladas por ciudadanos.</p> <ul style="list-style-type: none"> <li>• Usuario 3: Ciudadanos que consultan información publicada en el portal GVA (Uso Público) Ciudadanos que acceden al portal de la GVA para realizar consultas o trámites relacionados con la atención al ciudadano. Realizan consultas en lenguaje natural a través de un asistente inteligente o formularios de consulta.</li> </ul>	
<b>5</b>	<b>Usos finales previstos</b>
	<p>El uso previsto de la ontología es proporcionar un marco semántico que permita la creación de inteligencia en la Web mediante el uso de tecnologías de Web Semántica e Inteligencia Artificial. La ontología tiene como propósito estructurar, contextualizar y relacionar la información proveniente de la base de datos relacional Oracle, para hacerla accesible y comprensible por sistemas inteligentes que puedan operar sobre ella.</p> <p>La ontología debe ser capaz de representar el modelo de información del sistema de atención al ciudadano, sirviendo como un puente entre la base de datos relacional y las aplicaciones inteligentes que se desarrollen sobre ella: ayuda dinámica e inteligente que sustituya a la ayuda estática, detección de errores en el sistema a partir de la información ontológica y generación de listados personalizado en formato como Excel o CSV, de datos en base a consultas realizadas sobre la ontología.</p>
<b>6</b>	<b>Requisitos de la ontología</b>
	<p><b>a. Requisitos no Funcionales</b></p> <p>Las características técnicas y metodológicas que debe cumplir la ontología para asegurar su calidad, coherencia y escalabilidad son:</p> <ol style="list-style-type: none"> <li>6. Idioma y formato de la ontología: <ul style="list-style-type: none"> <li>○ La ontología se desarrolla en castellano, utilizando OWL (Web Ontology Language) y representada en formato RDF (Resource Description Framework).</li> </ul> </li> <li>7. Escalabilidad: <ul style="list-style-type: none"> <li>○ La ontología debe ser extensible para permitir la incorporación de nuevas clases, propiedades y relaciones sin comprometer su coherencia.</li> <li>○ Debe garantizar que la adición de nuevas áreas de conocimiento o módulos sea posible sin afectar negativamente el rendimiento del sistema.</li> </ul> </li> <li>8. Compatibilidad e interoperabilidad: <ul style="list-style-type: none"> <li>○ La ontología debe ser compatible con tecnologías de Web Semántica, permitiendo consultas SPARQL y su integración con sistemas externos mediante mapeo ontológico con la base de datos Oracle.</li> </ul> </li> <li>9. Rendimiento: <ul style="list-style-type: none"> <li>○ Las consultas realizadas sobre la ontología deben procesarse de manera eficiente, incluso con grandes volúmenes de datos.</li> <li>○ Debe garantizarse un rendimiento adecuado en escenarios de uso intensivo por múltiples usuarios.</li> </ul> </li> <li>10. Documentación y trazabilidad: <ul style="list-style-type: none"> <li>○ El proceso de construcción y actualización de la ontología debe estar claramente documentado mediante el documento Ontology Requirements Specification Document (ORSD).</li> </ul> </li> </ol>

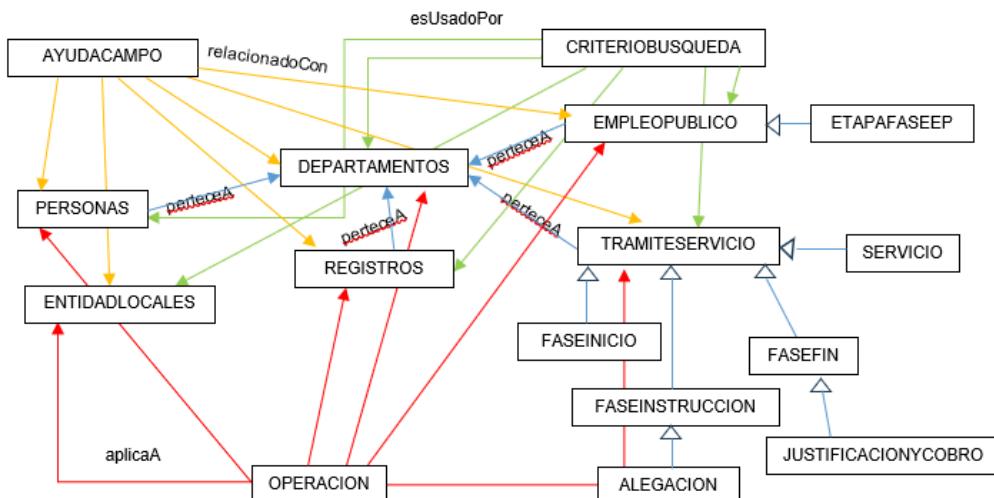
- Toda modificación debe registrarse para asegurar su trazabilidad.

#### b. Requisitos Funcionales

Ver: Tabla 1, Tabla 2, Tabla 3, Tabla 4, Tabla 5, Tabla 6, Tabla 7, Tabla 8, Tabla 9.

### 2.1.2 Conceptualización

A continuación, se muestra un primer mapa conceptual que representa los principales conceptos que deberá tener la red de ontologías, que describe el dominio de Atención al Ciudadano denominado como ontoAC. Para llevarlo a cabo, se han tenido en cuenta las preguntas de competencia y la especificación de requisitos descritas en los apartados anteriores, siguiendo las directrices metodológicas



propuestas por NeOn.

El mapa conceptual ofrece una visión global de las entidades clave y sus relaciones, estructurando de manera jerárquica y semánticamente coherente los elementos más representativos del dominio:

- Las clases principales representan los elementos fundamentales de la administración pública, como Personas, Departamentos, Registros, EntidadLocales, TramiteServicio o EmpleoPublico.
- Cada una de estas clases está vinculada mediante relaciones específicas que reflejan su comportamiento dentro del sistema. Por ejemplo, una Persona

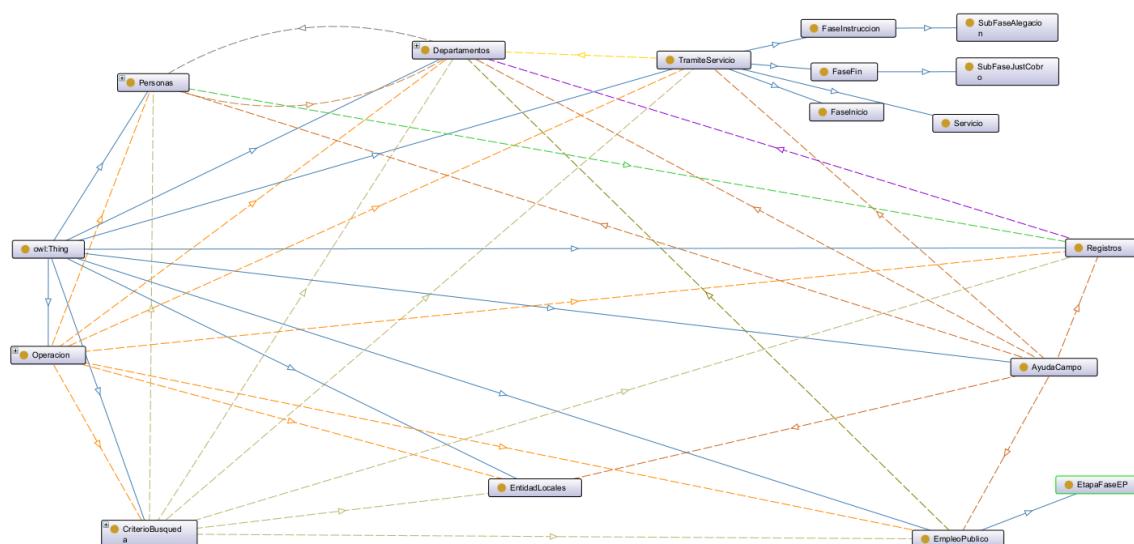
pertenece a un Departamento, y un TramiteServicio se compone de varias fases (FaseInicio, FaseInstruccion, FaseFin).

- Se incluyen también conceptos de soporte como CriterioBusqueda, Operacion y AyudaCampo, que permiten enriquecer la ontología con capacidades de interacción inteligente y flexibilidad para la búsqueda de información.
- Las subfases (Alegacion, JustificacionYCobro) se modelan como especializaciones dentro de las fases del procedimiento administrativo, garantizando la trazabilidad y el análisis detallado de cada etapa del proceso.

Este mapa conceptual actúa como base para la construcción de la ontología en OWL, asegurando que el modelo resultante sea comprensible, extensible y alineado con los objetivos del sistema. Además, permitirá definir con claridad las reglas de inferencia, las consultas SPARQL y los mecanismos de ayuda inteligente que formarán parte de la solución propuesta.

### 2.1.3 Implementación

La ontología ontoAC se ha implementado utilizando la herramienta Protégé<sup>2</sup> (Musen, 2015), siguiendo un enfoque modular y reutilizable. Las relaciones entre las entidades están definidas mediante propiedades objetuales y propiedades de datos, respetando el vocabulario específico del dominio y facilitando el mapeo posterior a la base de datos relacional Oracle.



Las clases del modelo son:

- *Personas*: Representa a las personas trabajadoras de la administración pública, con atributos como nombre, apellidos, cargo, despacho o código identificador.
- *Departamentos*: Estructuras administrativas a las que están adscritas personas, trámites y registros. Incluyen información sobre ubicación, normativa de creación, funciones, etc.
- *Registros*: Unidades que gestionan la recepción y salida de documentos en la administración. Pueden estar asociadas a departamentos.

<sup>2</sup>Stanford Center for Biomedical Informatics Research. *Protégé*. Disponible en: <https://protege.stanford.edu/>

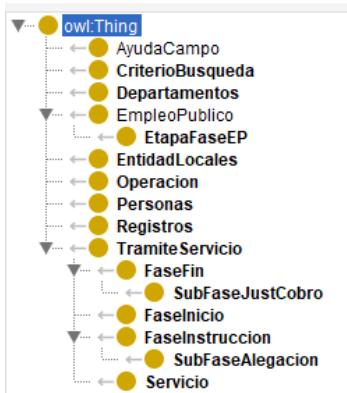
- *EntidadLocales*: Recoge información sobre ayuntamientos, diputaciones, mancomunidades y otras entidades territoriales.
- *TramiteServicio*: Representa los procedimientos y servicios administrativos ofrecidos al ciudadano. Incluye fases como *FaseInicio*, *FaseInstruccion*, *FaseFin*, y sus respectivas subfases, así como *Servicio*.
- *EmpleoPublico*: Contiene información sobre convocatorias y procesos selectivos de empleo público.
- *EtapaFaseEP*: Describe las distintas etapas dentro de un proceso de empleo público, como presentación de solicitudes o pruebas selectivas.
- *AyudaCampo*: Clase transversal que proporciona explicaciones inteligentes asociadas a los campos del sistema, con el objetivo de asistir al usuario.
- *Operacion*: Define las acciones posibles sobre cada módulo, como Consultar, Modificar, Insertar, Eliminar o Exportar.
- *CriterioBusqueda*: Estructura que define los criterios mediante los cuales el usuario puede realizar búsquedas personalizadas sobre los distintos módulos.

Las relaciones clave entre entidades más relevantes son:

- *ayudaCampoRelacionadoCon*  
Relaciona la entidad AyudaCampo con los campos o módulos del sistema sobre los que proporciona ayuda contextual e inteligente.

- *crbEsUsadoPor*  
Indica qué módulo (por ejemplo: Personas, Departamentos, etc.) hace uso de un determinado CriterioBusqueda.
- *depTienePersonas*  
Define la relación entre un Departamento y las Personas que tiene asignadas.
- *epPerteneceADep*  
Define la relación entre un Departamento y el empleo público.
- *opeAplicaA*  
Relaciona una Operacion (Consultar, Insertar, Modificar, Eliminar, Exportar) con el módulo del sistema al que puede aplicarse.
- *perPerteneceADep*  
Establece la adscripción de una persona a un departamento específico, reflejando la relación organizativa.
- *perPerteneceAReg*  
Relaciona una persona con un Registro administrativo en el que presta servicio o al que está vinculada.
- *regPerteneceADep*  
Indica qué Departamento es responsable de un determinado Registro.
- *trsPerteneceADep*  
Asocia un TrámiteServicio con el Departamento que lo gestiona.

La jerarquía de las clases en la ontología se visualiza en la siguiente imagen:



En el Anexo I se recogen capturas de pantalla representativas de la ontología implementada, que ilustran la estructura final de clases, propiedades y relaciones.

## 2.1.4 Evaluación

La evaluación de la ontología ontoAC se realiza con el objetivo de comprobar su calidad técnica y asegurar que cumple con los requisitos definidos previamente. Para ello, se han aplicado dos enfoques complementarios: *verificación* y *validación*, siguiendo buenas prácticas en Ingeniería Ontológica.

### Verificación

La verificación consiste en comprobar si la ontología ha sido desarrollada de forma correcta desde un punto de vista técnico y formal. Para ello, se ha comprobado la consistencia del modelo y aplicado la herramienta OOPS! <sup>1</sup>

---

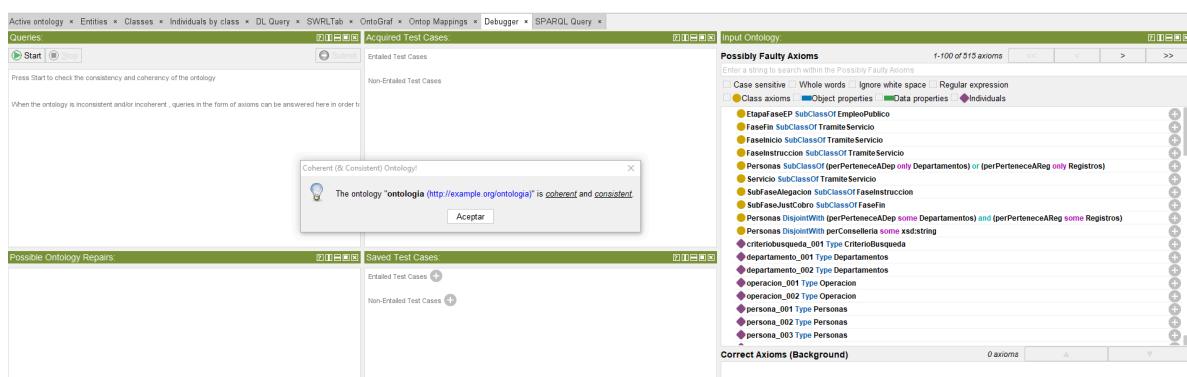
<sup>1</sup> <https://oops.linkeddata.es/>

#### *Comprobación de la consistencia*

Para comprobar la consistencia lógica de la ontología, se ha utilizado el razonador Pellet, disponible por defecto en la herramienta Protégè (versión 5.6.5), donde se desarrolló la ontología.

Pellet es un razonador open source desarrollado en Java que permite verificar ontologías definidas en OWL DL, asegurando que no existen contradicciones lógicas internas y que las definiciones de clases, propiedades y axiomas son coherentes.

La ejecución del razonador Pellet no reportó errores de consistencia ni conflictos entre axiomas, lo que confirma que la ontología es formalmente coherente y puede ser utilizada para inferencias seguras:



#### *Comprobación estructura de la ontología*

Como parte del proceso de verificación técnica, se ha realizado un análisis estructural de la ontología ontoAC utilizando la herramienta OOPS! (OntOlogy Pitfall

Scanner!), una plataforma reconocida para la detección automática de problemas comunes en el desarrollo de ontologías.

El objetivo de esta evaluación es identificar posibles *pitfalls* o errores de modelado que, aunque no siempre afectan directamente a la consistencia lógica, pueden comprometer la calidad, mantenibilidad y reutilización de la ontología.

OOPS! clasifica los resultados detectados en tres categorías según su gravedad:

- Menores (Minor): Recomendaciones de mejora que no constituyen errores, pero pueden optimizar la claridad o reutilización de la ontología.
- Importantes (Important): Aspectos que deben ser revisados, ya que pueden afectar a la comprensión, interoperabilidad o evolución futura del modelo.
- Críticos (Critical): Problemas que pueden comprometer la validez del razonamiento o la aplicabilidad de la ontología en entornos semánticos.

El análisis realizado sobre ontoAC obtuvo los siguientes resultados:

- **P05: Definición incorrecta de relaciones inversas.** Se detectó que depTienePersonas podría no ser inversa de perPerteneceADep. Es importante revisar si esta relación bidireccional está correctamente definida o si conviene modelarla de otra forma.
- **P19: Definición múltiple de dominios o rangos en propiedades.** Se hallaron 42 casos en los que una misma propiedad tiene múltiples dominios o rangos definidos, lo cual puede provocar ambigüedad y problemas de inferencia.
- **P24: Uso de definiciones recursivas.** Se detectaron 39 definiciones con recursividad, lo cual puede afectar a la claridad y al razonamiento automático si no está justificado adecuadamente.
- **P08: Faltan anotaciones.** Se identificaron 191 elementos sin anotaciones como etiquetas (`rdfs:label`) o comentarios (`rdfs:comment`), lo que dificulta la comprensión y reutilización.
- **P13: Relaciones inversas no declaradas explícitamente.** En 7 casos se utilizan relaciones que podrían beneficiarse de la declaración explícita de su inversa para mejorar la navegabilidad y el razonamiento.

Imagen primera iteración de resultados OOPS!

## Evaluation results

There are three levels of importance in pitfalls according to their impact on the ontology:

- **Critical** It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important** Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor** It is not really a problem, but by correcting it we will make the ontology nicer.

Pitfalls detected:

Results for P05: Defining wrong inverse relationships. 1 case	1 case	Critical
Two relationships are defined as inverse relations when they are not necessarily inverse.		
• This pitfall appears in the following elements: : <a href="http://example.org/ontologia#depTienePersonas">http://example.org/ontologia#depTienePersonas</a> may not be inverse of <a href="http://example.org/ontologia#perPerteneceADep">http://example.org/ontologia#perPerteneceADep</a>		
Results for P08: Missing annotations.	191 cases	Minor
Results for P13: Inverse relationships not explicitly declared.	7 cases	Minor
Results for P19: Defining multiple domains or ranges in properties.	42 cases	Critical
Results for P24: Using recursive definitions.	39 cases	Important

En base a los resultados se revisa la ontología y se vuelve a lanzar el scanner para evaluar:

Imagen segunda iteración de resultados OOPS!

## Evaluation results

There are three levels of importance in pitfalls according to their impact on the ontology:

- **Critical** It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important** Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor** It is not really a problem, but by correcting it we will make the ontology nicer.

Pitfalls detected:

Results for P13: Inverse relationships not explicitly declared.

23 cases

**Minor**

Se han corregido los siguientes pitfalls:

- P08 – Ausencia de anotaciones: Se han añadido anotaciones descriptivas en las clases y propiedades principales, mejorando la comprensión del modelo.
- P19 – Múltiples dominios o rangos en propiedades: Se han redefinido las propiedades afectadas, evitando el uso de múltiples dominios o rangos, de acuerdo con las buenas prácticas en OWL.
- P24 – Uso de definiciones recursivas: Se han reorganizado las definiciones afectadas para eliminar ciclos innecesarios o redundantes.
- En cuanto al pitfall P13 (relaciones inversas no declaradas), se ha optado por no realizar modificaciones, ya que:
  - Se trata de un pitfall de tipo menor que no afecta a la consistencia ni a la funcionalidad básica del modelo.
  - La ontología se ha diseñado para permitir inferencias directas, y la ausencia de relaciones inversas no compromete el objetivo planteado.
  - En caso de requerirse en versiones futuras, estas relaciones podríanadirse de forma controlada según las necesidades de navegación semántica.

Esta evaluación estructural ha permitido verificar que la ontología cumple con los criterios técnicos establecidos, y que no presenta errores críticos que puedan comprometer su uso dentro del sistema desarrollado.

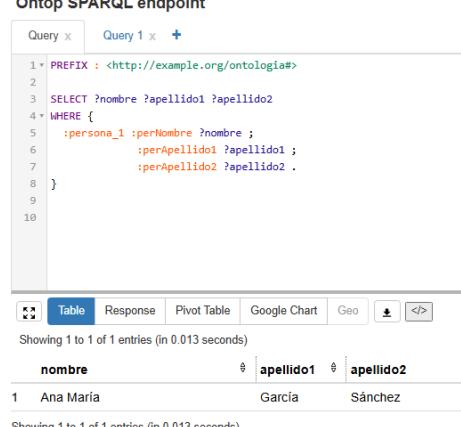
### Validación

La validación se centra en comprobar si la ontología desarrollada permite representar adecuadamente el conocimiento del dominio de Atención al Ciudadano, y si responde de manera efectiva a los requisitos prácticos definidos en la fase de especificación de requisitos.

#### *Comprobación preguntas de competencia*

Como parte del proceso de validación, se ha verificado que la ontología permite responder a las preguntas de competencia planteadas en el documento ORSD (Ontology Requirements Specification Document). Para ello, se han definido y ejecutado consultas SPARQL sobre el modelo ontológico, utilizando el motor Ontop conectado a la base de datos Oracle.

Las consultas han sido diseñadas para reflejar el tipo de información que se espera obtener en escenarios reales de uso del sistema. A continuación, se presenta evidencia de la ejecución de algunas de estas preguntas de forma aleatoria, con el objetivo de demostrar que la ontología cubre adecuadamente los conceptos, propiedades y relaciones necesarias:

Identificador	Pregunta de competencia	Resultado						
PER_1	¿Nombre y apellidos persona con código 1?	<p>Ontop SPARQL endpoint</p>  <pre> PREFIX : &lt;http://example.org/ontologia#&gt; SELECT ?nombre ?apellido1 ?apellido2 WHERE {   :persona_1 :perNombre ?nombre ;   :perApellido1 ?apellido1 ;   :perApellido2 ?apellido2 . } </pre> <p>Showing 1 to 1 of 1 entries (in 0.013 seconds)</p> <table border="1"> <thead> <tr> <th>nombre</th> <th>apellido1</th> <th>apellido2</th> </tr> </thead> <tbody> <tr> <td>Ana María</td> <td>García</td> <td>Sánchez</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.013 seconds)</p>	nombre	apellido1	apellido2	Ana María	García	Sánchez
nombre	apellido1	apellido2						
Ana María	García	Sánchez						

Identificador	Pregunta de competencia	Resultado
---------------	-------------------------	-----------

PER_24	¿Nombre de personas adscritas al departamento 101?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query x    Query 1 x    +</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?nombrePersona 4 WHERE { 5   ?persona :perPerteneceADep :departamento_101 . 6   ?persona :perNombre ?nombrePersona . 7 } 8 9 </pre> <p>Table Response Pivot Table Google Chart Geo &lt;/&gt;</p> <p>Showing 1 to 3 of 3 entries (in 0.036 seconds)</p> <table border="1"> <thead> <tr> <th>nombrePersona</th> </tr> </thead> <tbody> <tr> <td>1 Javier</td> </tr> <tr> <td>2 Ana María</td> </tr> <tr> <td>3 Raúl</td> </tr> </tbody> </table> <p>Showing 1 to 3 of 3 entries (in 0.036 seconds)</p>	nombrePersona	1 Javier	2 Ana María	3 Raúl
nombrePersona						
1 Javier						
2 Ana María						
3 Raúl						

Identificador	Pregunta de competencia	Resultado		
DEP_17	¿Rango departamento 101?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 8 x    Query 6 x    Query 7 x    Query x    Q</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?rango 4 WHERE { 5   :departamento_101 :depRango ?rango . 6 } 7 </pre> <p>Table Response Pivot Table Google Chart Geo</p> <p>Showing 1 to 1 of 1 entries (in 0.333 seconds)</p> <table border="1"> <thead> <tr> <th>rango</th> </tr> </thead> <tbody> <tr> <td>1 Dirección General</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.333 seconds)</p>	rango	1 Dirección General
rango				
1 Dirección General				

Identificador	Pregunta de competencia	Resultado
---------------	-------------------------	-----------

DEP_5	¿Horario departamento 101?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 9 x    Query 8 x    Query 6 x    Q</p> <pre> 1 PREFIX : &lt;http://example.org/ontologia# 2 PREFIX xsd: &lt;http://www.w3.org/2001/XMLSchema# 3 4 SELECT ?horario 5 WHERE { 6   :departamento_101 :depHorario ?horario 7 } 8 9 </pre> <p>Table Response Pivot Table Google</p> <p>Showing 1 to 1 of 1 entries (in 0.025 seconds)</p> <table border="1"> <thead> <tr> <th>horario</th> </tr> </thead> <tbody> <tr> <td>1 Lunes a Viernes 8:00 - 15:00</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.025 seconds)</p>	horario	1 Lunes a Viernes 8:00 - 15:00
horario				
1 Lunes a Viernes 8:00 - 15:00				

Identificador	Pregunta de competencia	Resultado		
CRB_1	¿Nombre criterio de búsqueda 1?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 6 x    Query 7 x    Query x    Query 1 x    Query 3 x    Query 2 x</p> <pre> 1 PREFIX : &lt;http://example.org/ontologia# 2 3 SELECT ?nombre 4 WHERE { 5   :criterioBusqueda_1 :crbCriteria ?nombre . 6 } </pre> <p>Table Response Pivot Table Google Chart Geo ↻</p> <p>Showing 1 to 1 of 1 entries (in 0.405 seconds)</p> <table border="1"> <thead> <tr> <th>nombre</th> </tr> </thead> <tbody> <tr> <td>1 Nombre Parcial</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.405 seconds)</p>	nombre	1 Nombre Parcial
nombre				
1 Nombre Parcial				

Identificador	Pregunta de competencia	Resultado
---------------	-------------------------	-----------

CRB_4	¿Uso comodín para el criterio de búsqueda 1?	<p><b>Ontop SPARQL endpoint</b></p> <pre>Query 6 x Query 7 x Query x Query 1 x Qu 1 * PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?uso 4 WHERE { 5   :criterioBusqueda_1 :crbUso ?uso . 6 } 7</pre> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p> <table border="1"> <thead> <tr> <th>uso</th> </tr> </thead> <tbody> <tr> <td>1 Ejemplo: Ana% devuelve Ana, Ana María, Ana Isabel.</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p>	uso	1 Ejemplo: Ana% devuelve Ana, Ana María, Ana Isabel.
uso				
1 Ejemplo: Ana% devuelve Ana, Ana María, Ana Isabel.				

Identificador	Pregunta de competencia	Resultado		
OPE_1	¿Nombre operación 21?	<p><b>Ontop SPARQL endpoint</b></p> <pre>Query 12 x Query 11 x Query 10 x Query 9 x Query 8 x 1 * PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?nombre 4 WHERE { 5   :operacion_21 :opeNombre ?nombre . 6 } 7</pre> <p>Showing 1 to 1 of 1 entries (in 0.019 seconds)</p> <table border="1"> <thead> <tr> <th>nombre</th> </tr> </thead> <tbody> <tr> <td>1 Consultar</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.019 seconds)</p>	nombre	1 Consultar
nombre				
1 Consultar				

Identificador	Pregunta de competencia	Resultado		
OPE_2	¿Descripción operación 1?	<p><b>Ontop SPARQL endpoint</b></p> <pre>Query 11 x Query 10 x Query 9 x Query 8 x Quer 1 * PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?descripcion 4 WHERE { 5   :operacion_21 :opeDescripcion ?descripcion . 6 } 7</pre> <p>Showing 1 to 1 of 1 entries (in 0.018 seconds)</p> <table border="1"> <thead> <tr> <th>descripcion</th> </tr> </thead> <tbody> <tr> <td>1 Permite visualizar información registrada en el sistema.</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.018 seconds)</p>	descripcion	1 Permite visualizar información registrada en el sistema.
descripcion				
1 Permite visualizar información registrada en el sistema.				

Identificador	Pregunta de competencia	Resultado								
OPE_3	¿Operaciones existentes?	<p>Ontop SPARQL endpoint</p> <p>Query 10 ×    Query 9 ×    Query 8 ×</p> <pre> 1+ PREFIX : &lt;http://example.org/onto1 2 3 SELECT ?nombre 4 WHERE { 5   ?operacion a :Operacion ; 6   :opNombre ?nombre . 7 } 8 ORDER BY ?operacion 9 10 11 </pre> <p>Showing 1 to 7 of 7 entries (in 0.058 seconds)</p> <table border="1"> <thead> <tr> <th>nombre</th> </tr> </thead> <tbody> <tr><td>1 Editar</td></tr> <tr><td>2 Buscar</td></tr> <tr><td>3 Consultar</td></tr> <tr><td>4 Insertar</td></tr> <tr><td>5 Modificar</td></tr> <tr><td>6 Eliminar</td></tr> <tr><td>7 Exportar</td></tr> </tbody> </table> <p>Showing 1 to 7 of 7 entries (in 0.058 seconds)</p>	nombre	1 Editar	2 Buscar	3 Consultar	4 Insertar	5 Modificar	6 Eliminar	7 Exportar
nombre										
1 Editar										
2 Buscar										
3 Consultar										
4 Insertar										
5 Modificar										
6 Eliminar										
7 Exportar										

Identificador	Pregunta de competencia	Resultado
---------------	-------------------------	-----------

REG_1	¿Nombre del registro 202?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 13 x    Query 12 x    Query 11 x    Qu</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?nombre 4 WHERE { 5   :registro_202 :regNombre ?nombre . 6 } 7 8 9 10 </pre> <p>Table Response Pivot Table Google Char</p> <p>Showing 1 to 1 of 1 entries (in 0.026 seconds)</p> <table border="1"> <thead> <tr> <th>nombre</th> </tr> </thead> <tbody> <tr> <td>1 Registro General</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.026 seconds)</p>	nombre	1 Registro General
nombre				
1 Registro General				

Identificador	Pregunta de competencia	Resultado		
TRA_6	¿Es un trámite o un servicio trámite código 1?	<p>Query 19 x +</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?tipo 4 WHERE { 5   :trámiteServicio_1 :trsTipo ?tipo . 6 } 7 </pre> <p>Table Response Pivot Table Google Char</p> <p>Showing 1 to 1 of 1 entries (in 0.022 seconds)</p> <table border="1"> <thead> <tr> <th>tipo</th> </tr> </thead> <tbody> <tr> <td>1 Trámite</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.022 seconds)</p>	tipo	1 Trámite
tipo				
1 Trámite				

Identificador	Pregunta de competencia	Resultado		
TRA_8	¿Código del trámite tipo servicio asignado al servicio?	<p>Ontop SPARQL endpoint</p> <p>Query 18 x    Query 16 x    Query 17 x    Query 15 x    Query 14 x    Q1</p> <p>Query 19 x    <b>Query 20 x</b> +</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?codigoTramite 4 WHERE { 5   :servicio_4 :servicioPerteneceA ?tramite . 6   BIND(STRAFTER(STR(?tramite), "tramiteServicio_") AS ?codigoTramite) 7 } 8 9 10 11 </pre> <p>Showing 1 to 1 of 1 entries (in 0.013 seconds)</p> <table border="1"> <thead> <tr> <th>codigoTramite</th> </tr> </thead> <tbody> <tr> <td>1 4</td> </tr> </tbody> </table>	codigoTramite	1 4
codigoTramite				
1 4				

Identificador	Pregunta de competencia	Resultado		
TRA_19	¿Plazo del servicio 4?	<p>Ontop SPARQL endpoint</p> <p>Query 18 x    Query 16 x    Query 17 x</p> <p>Query 19 x    Query 20 x    <b>Query 21 x</b> +</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?plazo 4 WHERE { 5   :servicio_4 :servPlazo ?plazo . 6 } 7 </pre> <p>Showing 1 to 1 of 1 entries (in 0.031 seconds)</p> <table border="1"> <thead> <tr> <th>plazo</th> </tr> </thead> <tbody> <tr> <td>1 15 días hábiles</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.031 seconds)</p>	plazo	1 15 días hábiles
plazo				
1 15 días hábiles				

Identificador	Pregunta de competencia	Resultado
TRA_21	¿Normativa de la fase inicio del trámite 1?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17 x    Query 18 x    Query 19 x    Query 20 x    Query 21 x</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?normativa 4 v WHERE { 5   ?faseInicio :finiPerteneceA :tramiteServicio_1 . 6   ?faseInicio :finiNorma ?normativa . 7 } 8 9 </pre> <p>Table Response Pivot Table Google Chart</p> <p>Showing 1 to 1 of 1 entries (in 0.048 seconds)</p> <p><b>normativa</b></p> <p>1 Ley 12/2025 de Procedimientos Administrativos.</p> <p>Showing 1 to 1 of 1 entries (in 0.048 seconds)</p>

Identificador	Pregunta de competencia	Resultado
TRA_32	¿Departamento de tramitación de la fase de instrucción del trámite 4?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17 x    Query 15 x    Query 16 x    Query 17 x    Query 18 x    Query 19 x    Query 20 x    Query 21 x</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?departamento 4 v WHERE { 5   ?faseInstruccion :finsPerteneceA :tramiteServicio_4 . 6   ?faseInstruccion :finsDeptTramita ?departamento . 7 } 8 </pre> <p>Table Response Pivot Table Google Chart Geo</p> <p>Showing 1 to 1 of 1 entries (in 0.026 seconds)</p> <p><b>departamento</b></p> <p>1 Departamento Técnico</p> <p>Showing 1 to 1 of 1 entries (in 0.026 seconds)</p>

Identificador	Pregunta de competencia	Resultado		
TRA_41	¿Objeto de la subfase de alegación de la fase instrucción 4?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17 x    Query 15 x    Query 1x      Query 24 x    Query 23 x    Query 22 x    Query 19 x    Query 2l</p> <pre> 1 * PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?objeto 4 WHERE { 5   ?subfase :sinsaPerteneceA :faseInstrucion_4 . 6   ?subfase :sinsaObjeto ?objeto . 7 } 8 </pre> <p>Showing 1 to 1 of 1 entries (in 0.035 seconds)</p> <p><b>objectId</b></p> <table border="1"> <tr> <td>1</td> <td>Ofrecer plazo para subsanar errores o aportar información adicional.</td> </tr> </table> <p>Showing 1 to 1 of 1 entries (in 0.035 seconds)</p>	1	Ofrecer plazo para subsanar errores o aportar información adicional.
1	Ofrecer plazo para subsanar errores o aportar información adicional.			

Identificador	Pregunta de competencia	Resultado		
TRA_44	¿Descripción de la fase de finalización del trámite 4?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17 x    Query 15 x      Query 25 x    Query 24 x    Query 23 x    Query 22 x</p> <pre> 1 * PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?descripcion 4 WHERE { 5   ?faseFin :ffinPerteneceA :tramiteServicio_4 . 6   ?faseFin :ffinDescripcion ?descripcion . 7 } 8 9 </pre> <p>Showing 1 to 1 of 1 entries (in 0.028 seconds)</p> <p><b>descripcion</b></p> <table border="1"> <tr> <td>1</td> <td>Finalización del procedimiento con resolución administrativa.</td> </tr> </table> <p>Showing 1 to 1 of 1 entries (in 0.028 seconds)</p>	1	Finalización del procedimiento con resolución administrativa.
1	Finalización del procedimiento con resolución administrativa.			

Identificador	Pregunta de competencia	Resultado		
TRA_51	¿Enlace de la subfase de justificación y cobro de la fase fin 4?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17 x      Query 26 x    Query 25 x    Query 24 x</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?enlace 4 v WHERE { 5   ?subfase :sffjcPerteneceA :faseFin_4 . 6   :faseFin_4 :ffinEnlace ?enlace . 7 } 8   </pre> <p>Showing 1 to 1 of 1 entries (in 0.052 seconds)</p> <p><b>enlace</b></p> <table border="1"> <tr> <td>1</td> <td>http://example.org/finalizacion-2</td> </tr> </table> <p>Showing 1 to 1 of 1 entries (in 0.052 seconds)</p>	1	http://example.org/finalizacion-2
1	http://example.org/finalizacion-2			

Identificador	Pregunta de competencia	Resultado		
TRA_55	¿Normativa de la subfase de justificación y cobro de la fase fin 4?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17 x    Query 18 x      Query 26 x    Query 27 x    Query 25 x    Query 24 x</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?normativa 4 v WHERE { 5   ?subfase :sffjcPerteneceA :faseFin_4 . 6   ?subfase :sffjcNorma ?normativa . 7 } 8   </pre> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p> <p><b>normativa</b></p> <table border="1"> <tr> <td>1</td> <td>Normativa de justificación administrativa del Consell.</td> </tr> </table> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p>	1	Normativa de justificación administrativa del Consell.
1	Normativa de justificación administrativa del Consell.			

Identificador	Pregunta de competencia	Resultado		
EP_2	¿Convocatoria empleo público 1?	<p>Ontop SPARQL endpoint</p> <p>Query 14 x    Query 4 x    Query 13 x    Query 12 x</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?convocatoria 4 v WHERE { 5   :empleoPublico_1 :epuConvocatoria ?convocatoria . 6 }</pre> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p> <table border="1"> <thead> <tr> <th>convocatoria</th> </tr> </thead> <tbody> <tr> <td>1 Convocatoria 2025 - Asesor Jurídico</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p>	convocatoria	1 Convocatoria 2025 - Asesor Jurídico
convocatoria				
1 Convocatoria 2025 - Asesor Jurídico				

Identificador	Pregunta de competencia	Resultado			
EP_21	¿Fases del empleo público 1?	<p>Ontop SPARQL endpoint</p> <p>Query 15 x    Query 14 x    Query 4 x    Query 13 x    Query 12 x</p> <pre> 1 v PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?fase 4 v WHERE { 5   ?etapa :epuePerteneceA :empleoPublico_1 ; 6     :epueFase ?fase . 7 } 8 9 10 11</pre> <p>Showing 1 to 2 of 2 entries (in 0.068 seconds)</p> <table border="1"> <thead> <tr> <th>fase</th> </tr> </thead> <tbody> <tr> <td>1 Fase 1</td> </tr> <tr> <td>2 Fase 2</td> </tr> </tbody> </table> <p>Showing 1 to 2 of 2 entries (in 0.068 seconds)</p>	fase	1 Fase 1	2 Fase 2
fase					
1 Fase 1					
2 Fase 2					

Identificador	Pregunta de competencia	Resultado		
EP_22	¿Nombre de la fase1 del empleo público 1?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 16 x    Query 15 x    Query 14 x    Query 13 x</p> <pre> 1 * PREFIX : &lt;http://example.org/ontologia# 2 3 SELECT ?nombre 4 WHERE { 5   ?etapa :epuePerteneceA :empleoPublico_1 ; 6     :epueFase "Fase 1" ; 7     :epueNombre ?nombre . 8 } 9 </pre> <p>Showing 1 to 1 of 1 entries (in 0.043 seconds)</p> <table border="1"> <thead> <tr> <th>nombre</th> </tr> </thead> <tbody> <tr> <td>1 Presentación de Solicitudes</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.043 seconds)</p>	nombre	1 Presentación de Solicitudes
nombre				
1 Presentación de Solicitudes				

Identificador	Pregunta de competencia	Resultado		
EEL_1	¿Tipo entidad local 2?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 16 x    Query 17 x    Query 15 x</p> <pre> 1 * PREFIX : &lt;http://example.org/ontolo... 2 3 SELECT ?tipo 4 WHERE { 5   :entidadLocal_2 :enlTipo ?tipo . 6 } 7 </pre> <p>Showing 1 to 1 of 1 entries (in 0.023 seconds)</p> <table border="1"> <thead> <tr> <th>tipo</th> </tr> </thead> <tbody> <tr> <td>1 Mancomunidad</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (in 0.023 seconds)</p>	tipo	1 Mancomunidad
tipo				
1 Mancomunidad				

Identificador	Pregunta de competencia	Resultado	
EEL_4	¿Dirección entidad local 2?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query 18 x    Query 16 x    Query 17</p> <pre> 1 PREFIX : &lt;http://example.org/onto 2 3 SELECT ?direccion 4 WHERE { 5   :entidadLocal_2 :enlDireccion ?_ 6 } 7 </pre> <p>Table Response Pivot Table C</p> <p>Showing 1 to 1 of 1 entries (in 0.021 seconds)</p> <p><b>direccion</b></p> <table border="1"> <tr> <td>1 Calle de las Mancomunidades 23</td> </tr> </table> <p>Showing 1 to 1 of 1 entries (in 0.021 seconds)</p>	1 Calle de las Mancomunidades 23
1 Calle de las Mancomunidades 23			

Identificador	Pregunta de competencia	Resultado	
AYU_1	¿Campo 1?	<p><b>Ontop SPARQL endpoint</b></p> <p>Query x    Query 1 x    Query 2 x    Query 3 x    +</p> <pre> 1 PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?codigo 4 WHERE { 5   :ayudaCampo_1 :ayuCodigo ?codigo . 6 } 7 </pre> <p>Table Response Pivot Table Google Chart Geo C</p> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p> <p><b>codigo</b></p> <table border="1"> <tr> <td>1 PER_NOMBRE</td> </tr> </table> <p>Showing 1 to 1 of 1 entries (in 0.024 seconds)</p>	1 PER_NOMBRE
1 PER_NOMBRE			

Identificador	Pregunta de competencia	Resultado
AYU_3	¿Descripción del campo 1?	<p>Ontop SPARQL endpoint</p> <pre> 1+ PREFIX : &lt;http://example.org/ontologis#&gt; 2 3 SELECT ?descripcion 4 WHERE { 5   :ayudaCampo_1 :ayuNombre ?descripcion . 6 } 7 </pre> <p>Showing 1 to 1 of 1 entries (in 0.444 seconds)</p> <p><b>descripcion</b></p> <p>1 Introduzca el nombre completo de la persona tal y como aparece en su documento de identidad. No se permiten apodos ni abreviaturas</p> <p>Showing 1 to 1 of 1 entries (in 0.444 seconds)</p>

Identificador	Pregunta de competencia	Resultado
AYU_7	¿Validación del campo 1?	<p>Ontop SPARQL endpoint</p> <pre> 1+ PREFIX : &lt;http://example.org/ontologia#&gt; 2 3 SELECT ?validacion 4 WHERE { 5   :ayudaCampo_1 :ayuReglas ?validacion . 6 } 7 </pre> <p>Showing 1 to 1 of 1 entries (in 0.034 seconds)</p> <p><b>validacion</b></p> <p>1 Debe contener solo letras y espacios. Máximo 50 caracteres.</p> <p>Showing 1 to 1 of 1 entries (in 0.034 seconds)</p>

## 2.2 Arquitectura del Sistema basado en Ontología

Este apartado describe la arquitectura técnica implementada para integrar la ontología del dominio “Atención al Ciudadano” con una base de datos relacional, permitiendo la realización de consultas semánticas sobre datos reales. La solución se basa en el uso del motor Ontop para el mapeo semántico y la ejecución de consultas SPARQL, Oracle como sistema gestor de base de datos, y un backend desarrollado en Spring Boot encargado de la gestión de las peticiones, el procesamiento de resultados y la interacción con el frontend del sistema.

Además, la arquitectura incorpora el uso de un modelo de lenguaje (LLM) a través de la API de OpenAI, que permite transformar las preguntas formuladas por el usuario en lenguaje natural en consultas SPARQL válidas. Esta integración extiende la funcionalidad del sistema, haciéndolo más accesible y potente, y forma parte esencial del flujo de interacción entre usuario, conocimiento semántico y datos reales.

### 2.2.1 Componentes principales

El sistema desarrollado se apoya en varios componentes tecnológicos clave, que permiten integrar la ontología del dominio con una base de datos relacional y ejecutar consultas semánticas a partir de lenguaje natural. A continuación, se describen los principales elementos involucrados en la arquitectura, junto con su función específica y referencia técnica:

#### Ontop<sup>2</sup>

Es un motor de consultas SPARQL diseñado para trabajar directamente con bases de datos relacionales como Oracle, sin necesidad de duplicar los datos. En lugar de almacenar la información en un triplestore. Permite que las consultas semánticas (SPARQL) se ejecuten sobre los datos existentes en la base de datos, gracias a un mecanismo de mapeo conocido como OBDA (Ontology-Based Data Access).

Este mapeo se define en un archivo .obda, donde se especifica cómo los conceptos de la ontología (como Persona, Departamento, Trámite, etc.) se corresponden con las tablas y columnas reales de la base de datos Oracle. Este archivo actúa como un puente entre el modelo conceptual representado en la ontología y los datos almacenados en el sistema tradicional.

<sup>2</sup> <https://ontop-vkg.org>

Ontop desempeña un papel clave en el sistema, ya que permite que los usuarios realicen consultas en lenguaje SPARQL (más intuitivo y alineado con el modelo ontológico) y que estas se traduzcan automáticamente a sentencias SQL comprensibles por la base de datos Oracle. Este proceso es completamente transparente para el usuario final. Por ejemplo, si un usuario desea obtener un listado de personas de un departamento determinado, puede formular la pregunta en lenguaje natural, que luego es traducida a SPARQL. Ontop se encarga de transformar esa consulta en SQL, ejecutarla en Oracle y devolver los resultados estructurados según la ontología.

De este modo, Ontop permite aprovechar el poder de las consultas semánticas sin alterar la base de datos original ni duplicar la información.

#### OpenAI API (GPT)<sup>3</sup>:

La API de OpenAI permite utilizar modelos de lenguaje avanzados, como GPT, que son capaces de entender y generar texto en lenguaje natural. En el contexto de este sistema, se ha configurado con un *prompt* específico que traduce las preguntas formuladas por el usuario en lenguaje natural (por ejemplo, "¿Qué personas pertenecen al departamento de Juventud?") a consultas SPARQL válidas, que siguen la estructura y términos definidos en la ontología del dominio de atención al ciudadano.

Cuando el usuario escribe una consulta o solicitud, esta se envía a la API de OpenAI, que genera automáticamente la consulta SPARQL correspondiente. Esta consulta luego se ejecuta sobre la base de datos Oracle a través de Ontop. Este enfoque permite que cualquier usuario, sin conocimientos técnicos, pueda interactuar con el sistema de forma natural y obtener respuestas personalizadas basadas en los datos reales.

#### Spring Boot (Java)<sup>4</sup>:

Spring Boot es un framework de desarrollo en Java que permite crear aplicaciones web de forma rápida, modular y escalable. Está especialmente diseñado para facilitar la creación de servicios backend, es decir, la parte del sistema que gestiona la lógica del servidor y la comunicación entre los distintos componentes.

---

<sup>3</sup> <https://platform.openai.com/docs>

<sup>4</sup> <https://spring.io/projects/spring-boot>

Spring Boot actúa como el núcleo del backend. Se encarga de gestionar las solicitudes HTTP que llegan desde el frontend (por ejemplo, cuando un usuario hace una pregunta o pulsa el botón de ayuda), y coordina la interacción con los otros módulos del sistema:

- Se conecta con la API de OpenAI para obtener la consulta SPARQL generada.
- Lanza la consulta a Ontop, que la traduce y ejecuta sobre Oracle.
- Recibe los resultados y los transforma en un formato comprensible (por ejemplo, JSON) para devolverlos al usuario.

Haciendo uso de Spring Boot, todo el flujo se realiza de forma automática, rápida y estructurada.

#### Oracle Database<sup>5</sup>

Oracle es un sistema gestor de bases de datos relacional ampliamente utilizado en entornos empresariales y administrativos. Se encarga de almacenar y organizar grandes volúmenes de datos estructurados, garantizando su integridad, disponibilidad y seguridad.

En este proyecto, Oracle es la fuente principal de datos reales. Aquí se encuentran almacenadas las tablas con información relevante del dominio, como personas, departamentos, registros administrativos, trámites, etc.

Aunque las consultas del usuario se formulan en lenguaje SPARQL, estas se traducen automáticamente a SQL (el lenguaje que entiende Oracle) gracias al motor Ontop. De esta forma, se pueden obtener respuestas actualizadas y basadas directamente en los datos reales del sistema, sin necesidad de migrarlos ni duplicarlos.

#### Ontología OWL (Web Ontology Language)<sup>6</sup>

Una ontología en OWL (Web Ontology Language) es un modelo conceptual que permite representar de forma estructurada y lógica el conocimiento de un dominio concreto. En este caso, la ontología ha sido diseñada para describir con precisión todos los elementos clave del sistema de atención al ciudadano: personas, departamentos, trámites, servicios, criterios de búsqueda, operaciones posibles, etc.

<sup>5</sup> <https://www.oracle.com/database>

<sup>6</sup> <https://www.w3.org/OWL/>

La ontología funciona como el vocabulario central del sistema: define qué entidades existen, cómo se relacionan entre sí y qué propiedades tiene cada una. Este modelo guía la interpretación semántica de las consultas SPARQL, asegurando que se entiendan correctamente según el contexto del dominio. Además, garantiza que las respuestas obtenidas sean coherentes y alineadas con la lógica y estructura del sistema administrativo.

Es también una pieza fundamental para facilitar la integración con herramientas de IA y permitir que los usuarios interactúen con el sistema en lenguaje natural, sin necesidad de conocer los detalles técnicos de la base de datos.

### 2.2.2 Modelo de datos en Oracle

El prototipo desarrollado incorpora un modelo de base de datos relacional diseñado específicamente para representar las entidades definidas en la ontología del dominio. Este modelo se ha construido con un enfoque funcional, abarcando las estructuras mínimas necesarias para cubrir los principales casos de uso del sistema y permitir la validación del enfoque ontológico propuesto.

A continuación, se describen las principales tablas del modelo, agrupadas por su funcionalidad dentro del sistema:

- *Gestión de personas y estructura organizativa:* Incluye tablas como AC\_PERSONAS, AC\_DEPARTAMENTOS y AC\_REGISTROS, que almacenan información básica sobre las personas registradas, los departamentos organizativos y los puntos de registro del sistema. Estas entidades representan la base estructural del dominio, permitiendo identificar quiénes forman parte del sistema, cómo se organiza la administración y desde dónde se prestan los servicios.
- *Trámites y servicios administrativos:* Las tablas AC\_TRAMITES\_SERVICIOS, AC\_SERVICIO, y las distintas fases (AC\_FASE\_INICIO, AC\_FASE\_INSTRUCCION, AC\_FASE\_FIN, etc.) reflejan el ciclo de vida completo de un procedimiento administrativo. Permiten modelar la información relacionada con los trámites disponibles, sus objetivos, destinatarios, requisitos, plazos, fases y subfases específicas como alegaciones o justificación de cobro.

- *Procesos de empleo público:* A través de las tablas AC\_EMPLEO\_PUBLICO y AC\_ETAPA\_FASE\_EP, se representa la estructura de convocatorias, pruebas, etapas y requisitos asociados a procesos de selección de personal. Este módulo permite gestionar información detallada sobre oportunidades de empleo dentro del sistema.
- *Criterios de búsqueda y ayuda contextual:* Las tablas AC\_CRITERIOS\_BUSQUEDA, AC\_CRB\_USADO\_POR y AC\_AYUDA\_CAMPO permiten definir los criterios de búsqueda disponibles en el sistema, sus relaciones con los distintos campos y la información de ayuda asociada. Estas estructuras son clave para habilitar funcionalidades como la ayuda inteligente y las consultas en lenguaje natural.
- *Entidades auxiliares y operaciones:* Tablas como AC\_ENTIDAD\_LOCAL, AC\_OPERACION y AC\_OPE\_APlica\_A proporcionan soporte a funcionalidades transversales del sistema, como la asociación de operaciones específicas a secciones concretas, o la representación de entidades locales vinculadas a procedimientos.

Cada tabla se ha diseñado para reflejar de manera coherente las entidades y relaciones establecidas en la ontología, facilitando el mapeo semántico mediante el archivo .obda y permitiendo la ejecución de consultas SPARQL sobre datos reales a través del motor Ontop.

El modelo de datos se incluye en el Anexo II del presente documento.

### **2.2.3 Flujo de interacción**

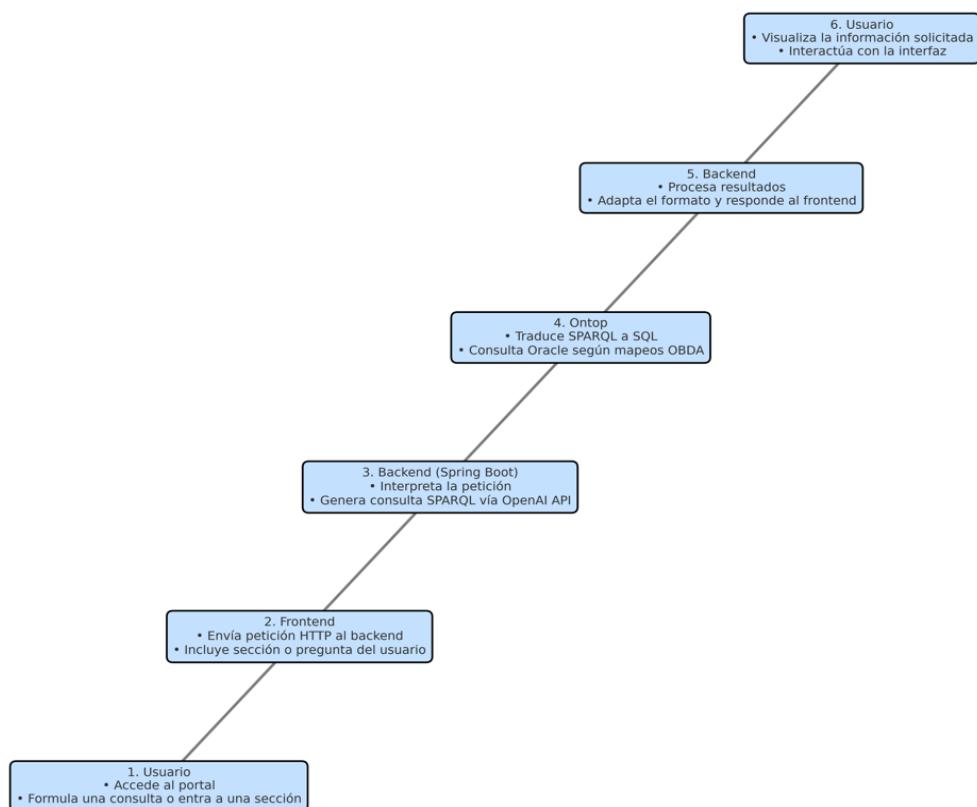
El flujo general de interacción entre los componentes del sistema es el siguiente:

1. El usuario accede a una sección del portal o formula una consulta en lenguaje natural, relacionada con algún elemento del dominio (por ejemplo, trámites, personas o servicios).
2. El frontend envía una petición al backend, indicando la consulta específica introducida por el usuario o la sección en la que se encuentra.
3. El backend genera o recupera una consulta SPARQL correspondiente al contenido solicitado, a través de la transformación de lenguaje natural utilizando la API de OpenAI.
4. Ontop recibe la consulta SPARQL y la traduce a SQL, accediendo a los datos reales almacenados en Oracle, según los mapeos definidos en el archivo .obda.

5. El resultado de la consulta se devuelve al backend, que lo procesa, adapta su formato si es necesario, y lo envía de vuelta al frontend.
6. El usuario visualiza la información solicitada en la interfaz.

La siguiente imagen muestra el flujo completo del sistema:

Flujo de interacción del sistema basado en ontología. Elaboración propia.



#### 2.2.4 Comunicación entre módulos

La arquitectura del sistema se ha diseñado siguiendo un enfoque modular y desacoplado, permitiendo que cada componente interactúe con los demás mediante interfaces bien definidas y protocolos estandarizados. La comunicación entre los distintos módulos se establece de la siguiente manera:

- Spring Boot -> Ontop: La aplicación backend, desarrollada en Spring Boot, se comunica con Ontop mediante peticiones HTTP al endpoint SPARQL expuesto por Ontop. Esta interacción permite ejecutar consultas semánticas generadas a

partir del lenguaje natural o predefinidas sobre la ontología y los datos almacenados en Oracle.

- Ontop -> Oracle: Ontop actúa como middleware semántico, estableciendo la conexión con la base de datos Oracle a través de JDBC. Esta conexión está configurada en los archivos de propiedades (.properties) y definida mediante los archivos de mapeo OBDA, que permiten traducir las consultas SPARQL en consultas SQL sobre la base de datos relacional.
- Frontend -> Spring Boot: El módulo frontend se comunica con el backend a través de endpoints REST. Esta interacción permite al usuario acceder a la ayuda inteligente contextual y realizar consultas personalizadas, que son gestionadas por los servicios del backend y delegadas a Ontop cuando es necesario.

Este diseño modular asegura la escalabilidad, reutilización y facilidad de mantenimiento del sistema, permitiendo la evolución futura de cada componente de forma independiente.

#### **2.2.5 Lógica de diseño**

El diseño arquitectónico adoptado en este sistema responde a la necesidad de construir una solución flexible, escalable y centrada en el conocimiento, que permita mejorar la experiencia de los usuarios sin comprometer la integridad ni la eficiencia del sistema original. En particular, esta arquitectura presenta las siguientes ventajas clave:

- Reutilización de los datos existentes: Gracias al uso de mapeos OBDA (Ontology-Based Data Access) mediante Ontop, se evita la duplicación de datos. La información se consulta directamente desde la base de datos relacional (Oracle), manteniendo su integridad y reduciendo costes de mantenimiento.
- Enriquecimiento semántico: La incorporación de una capa semántica basada en ontologías permite estructurar el conocimiento del dominio, aplicar reglas de inferencia y mejorar la comprensión de los conceptos implicados. Esto proporciona una base sólida para implementar funcionalidades inteligentes.
- Interacción inteligente con el usuario: La arquitectura habilita la implementación de una ayuda contextual, consultas personalizadas en lenguaje natural y exportación de resultados, generando una experiencia más intuitiva, accesible y eficiente para el usuario final.

- Facilidad de mantenimiento y evolución: Al estar basada en módulos desacoplados, la arquitectura permite modificar o ampliar la ontología, así como adaptar el modelo de datos subyacente, sin necesidad de rediseñar el sistema completo. Esta característica asegura la sostenibilidad a largo plazo del desarrollo.

#### **2.2.6 Fundamentos del uso exclusivo de Ontop como motor SPARQL**

Durante las fases iniciales del desarrollo del sistema, se valoró la posibilidad de adoptar un enfoque híbrido que integrase dos motores SPARQL diferenciados:

- Ontop, para acceder a datos reales almacenados en la base de datos relacional Oracle mediante mapeos OBDA.
- Apache Jena Fuseki, para realizar consultas semánticas sobre una ontología cargada como grafo RDF, aprovechando su razonador interno.

#### Evaluación del enfoque híbrido

Aunque esta arquitectura dual parecía idónea desde un punto de vista conceptual, su implementación práctica reveló diversas dificultades que comprometían la simplicidad, el mantenimiento y la coherencia del sistema:

- Complejidad de mantenimiento: La gestión de dos endpoints SPARQL, junto con la necesidad de mantener dos ontologías (una conectada a datos reales y otra almacenada como RDF), implicaba una carga técnica considerable.
- Problemas de integración: Se requería implementar una lógica capaz de discernir, para cada consulta en lenguaje natural, si debía dirigirse a Ontop o a Fuseki, algo que no siempre es trivial ni fiable.
- Riesgo de inconsistencia: Al disponer de datos en dos fuentes distintas, aumentaba la probabilidad de presentar información duplicada, desactualizada o incoherente para el usuario final.
- Complejidad innecesaria para el usuario: Desde el punto de vista del usuario, no resulta relevante si la información procede de una base de datos relacional o de un grafo RDF. Lo fundamental es que los resultados sean correctos, comprensibles y accesibles mediante lenguaje natural.

#### Decisión tomada: simplificación mediante Ontop

Considerando estos factores, y siguiendo el principio de simplicidad arquitectónica y mantenibilidad a largo plazo, se optó por un diseño unificado en el que:

- Todas las consultas SPARQL, incluidas las generadas automáticamente a partir de lenguaje natural, se ejecutan exclusivamente a través de Ontop, que actúa como capa de virtualización semántica sobre la base de datos Oracle.
- La ontología utilizada en Ontop no solo modela la estructura de los datos, sino que también incorpora anotaciones semánticas como rdfs:label, rdfs:comment y otros metadatos que pueden ser mapeados directamente desde Oracle si es necesario.

### **2.2.7 Elección de OpenAI para la generación de consultas SPARQL**

Uno de los objetivos principales del sistema desarrollado es permitir que los usuarios puedan realizar consultas personalizadas en lenguaje natural, sin necesidad de conocer la sintaxis SPARQL ni la estructura interna de la ontología. Esto implica que, desde el frontend, el usuario debe poder formular una pregunta libre en texto natural, y que el sistema sea capaz de interpretarla, traducirla a una consulta SPARQL válida y ejecutarla sobre los datos reales almacenados en Oracle, a través del mapeo semántico de Ontop.

Con este objetivo, se han analizado varias estrategias posibles para la generación de consultas SPARQL automáticas:

- Consultas SPARQL predefinidas: altamente precisas y eficientes, pero poco flexibles, ya que solo permiten responder a un conjunto limitado de preguntas. No permiten entrada libre por parte del usuario.
- Plantillas SPARQL parametrizadas: permiten cierto grado de flexibilidad, pero requieren que el usuario utilice formularios estructurados en lugar de lenguaje natural.
- Constructores de consultas (Query Builder): ofrecen una interfaz gráfica amigable, pero no permiten preguntas abiertas, sino únicamente combinaciones predefinidas de filtros.
- Modelos NLP entrenados específicamente (como gAnswer, Sparqling o BERT-to-SPARQL): permiten traducir lenguaje natural a SPARQL, pero requieren configuración y entrenamiento adaptado a la ontología del dominio. Son opciones viables, aunque su implementación es más compleja y escapa al alcance inmediato del proyecto.
- Modelos de lenguaje generativos (como OpenAI): permiten generar consultas SPARQL a partir de preguntas formuladas en lenguaje natural libre, siempre

que se les proporcione un contexto adecuado sobre la ontología utilizada. No requieren entrenamiento específico y permiten una integración directa mediante API.

De todas las opciones analizadas, solo aquellas basadas en modelos de lenguaje natural permiten satisfacer completamente el requisito funcional del sistema. En particular, OpenAI ofrece una solución inmediata, viable y funcional, permitiendo traducir preguntas formuladas por los usuarios en lenguaje natural a consultas SPARQL válidas, ejecutables directamente sobre el endpoint expuesto por Ontop.

Por tanto, se ha optado por integrar OpenAI en la arquitectura del sistema como componente generador de SPARQL. Este enfoque aporta flexibilidad y usabilidad al sistema sin comprometer la capacidad de ejecución sobre los datos reales. Como mejora futura, se contempla la posibilidad de sustituir o complementar este componente con modelos entrenados específicamente sobre la ontología del dominio, en caso de que se desee mayor independencia tecnológica o mayor control semántico sobre las consultas generadas.

# Parte 3

## 3. Desarrollo de la Ayuda Inteligente

En este apartado se describe el proceso completo de desarrollo del sistema de ayuda inteligente integrado en el portal de atención al ciudadano. El objetivo principal ha sido transformar la ayuda estática tradicional en un componente dinámico, contextual e interactivo, capaz de proporcionar respuestas útiles basadas en los datos reales del sistema.

Para ello, se ha seguido una metodología estructurada que abarca desde el análisis del dominio y el diseño de la ontología, hasta la integración técnica del botón de ayuda en la interfaz de usuario. Este componente modular permite al ciudadano obtener, en tiempo real, información precisa y personalizada a partir de preguntas formuladas en lenguaje natural.

El sistema se apoya en una ontología del dominio que estructura el conocimiento disponible, y utiliza consultas SPARQL ejecutadas sobre Ontop para acceder a los datos almacenados en una base de datos Oracle. Gracias a este enfoque, es posible ofrecer una ayuda inteligente y contextualizada, adaptada a cada sección del sistema.

Finalmente, se presenta una evaluación de la solución implementada, que incluye pruebas funcionales y métricas de rendimiento. Estos resultados permiten valorar la efectividad, precisión y utilidad de la ayuda inteligente en comparación con el sistema anterior.

### 3.1 Metodología

El desarrollo del sistema de ayuda inteligente se ha llevado a cabo mediante una metodología modular y basada en el conocimiento, apoyada en los principios de la Web Semántica y la reutilización de información estructurada. El objetivo ha sido sustituir la ayuda estática tradicional por un componente dinámico, capaz de ofrecer información contextualizada y vinculada a los elementos reales del sistema.

El proceso ha seguido un enfoque iterativo, estructurado en las siguientes fases:

1. *Análisis de la ayuda estática existente*: Se han identificado las limitaciones del modelo actual, basado en páginas HTML, que se visualizan al pulsar sobre botones estáticos y repetitivos en la aplicación, lo cual dificulta su mantenimiento, impide la contextualización dinámica y puede generar

confusión en el usuario. El impacto de estas deficiencias en la experiencia del usuario y en la eficiencia del sistema se ha analizado en detalle en el apartado *1.1 Motivación*, donde se justifica la necesidad de transformar la ayuda estática en un sistema inteligente y contextualizado.

2. *Modelado del conocimiento en una ontología*: Se ha definido una ontología del dominio “Atención al Ciudadano”, que representa las entidades, secciones, campos, criterios de búsqueda y operaciones más relevantes del sistema.

Esta ontología actúa como núcleo semántico que permite vincular cada elemento funcional con su correspondiente definición, requisitos o ejemplos de uso. Cabe señalar que la ontología utilizada corresponde a un prototipo acotado, diseñado específicamente para los fines de este trabajo.

La implementación completa del sistema requeriría un modelo de mayor envergadura, lo cual excede el alcance definido para este TFM.

Todo el proceso de modelado se ha detallado en el apartado *2.1 Creación de la Ontología del Dominio “Atención al Ciudadano” (ontoAC)*.

3. *Diseño del sistema de consultas SPARQL*: A partir del modelo ontológico, se ha desarrollado un sistema capaz de generar consultas SPARQL adaptadas al dominio de atención al ciudadano. Para ello, se han definido consultas preconfiguradas para escenarios recurrentes, y adicionalmente se ha integrado la API de OpenAI, que permite transformar preguntas formuladas en lenguaje natural en consultas SPARQL válidas. Este diseño se ha explicado con detalle en el apartado *2.2 Arquitectura del Sistema basado en Ontología*.

Esta funcionalidad ha permitido incorporar flexibilidad al sistema, facilitando la recuperación de información incluso cuando el usuario no conoce la estructura exacta de los datos o los términos definidos en la ontología.

Se ha creado un prompt predefinido para guiar al modelo de lenguaje en la generación de consultas SPARQL válidas a partir de preguntas en lenguaje natural. Este prompt incluye ejemplos representativos y restricciones sintácticas adaptadas al dominio modelado en la ontología. Ver apartado *3.2.4 Metodología de diseño del prompt para la generación de consultas SPARQL*.

4. *Implementación del backend modular*: Se ha desarrollado un servicio en Spring Boot que actúa como intermediario entre la interfaz de usuario, el motor de consultas Ontop y la base de datos Oracle.

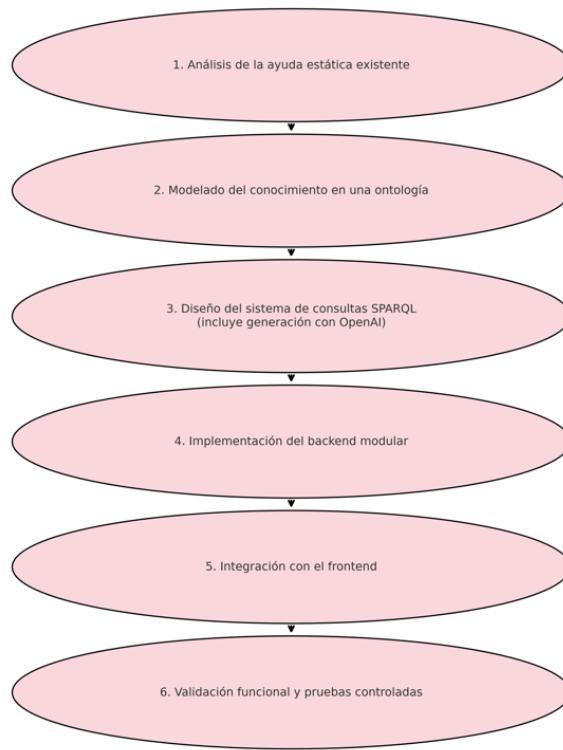
Este servicio recibe las preguntas formuladas en lenguaje natural desde el frontend, invoca la API de OpenAI para generar la consulta SPARQL correspondiente y la ejecuta sobre Ontop. Los resultados obtenidos son procesados y devueltos al frontend en formato estructurado para su presentación al usuario. Ver apartado 3.2.2 *Arquitectura del backend en Spring Boot*.

5. *Integración con el frontend*: Se ha diseñado un botón de ayuda contextual, que permite solicitar la ayuda en tiempo real sin necesidad de recargar la página ni abandonar el proceso que está realizando el usuario. Ver apartado 3.2.2 *Implementación del frontend y visualización de resultados*.
6. *Validación funcional y pruebas controladas*: Finalmente, se han realizado pruebas en un entorno de desarrollo para verificar el correcto funcionamiento del sistema, asegurando que la información proporcionada por la ayuda inteligente corresponde con exactitud al contexto en el que se solicita. Ver apartado 3.3 *Evaluación y validación*.

La imagen siguiente muestra de forma esquemática las fases que han compuesto el proceso metodológico aplicado en el desarrollo de la ayuda inteligente:

Imagen 3.1. Elaboración propia

#### Fases del desarrollo de la Ayuda Inteligente



## 3.2 Implementación del Botón de Ayuda Inteligente

El botón de ayuda modular tiene como objetivo proporcionar al usuario información útil y precisa relacionada con el dominio de atención al ciudadano, a partir de consultas realizadas en lenguaje natural. Este componente sustituye la ayuda estática tradicional por un sistema dinámico, interactivo y conectado a datos reales, mejorando la experiencia del usuario sin necesidad de abandonar el proceso en curso.

Se ha desarrollado como un módulo reutilizable, capaz de integrarse en cualquier parte del sistema sin necesidad de replicar código. Esto permite su despliegue en múltiples pantallas o formularios, manteniendo una lógica centralizada que gestiona el acceso al conocimiento semántico disponible en la ontología del dominio.

El usuario interactúa con el sistema a través del botón de ayuda, que abre un componente modal en la interfaz. Desde ahí, puede introducir preguntas en lenguaje natural relacionadas con cualquier aspecto del sistema, como trámites, personas, servicios o estructuras administrativas.

Estas consultas son enviadas desde el frontend al backend mediante una petición HTTP al endpoint `/api/ayuda/`. El backend, desarrollado en Spring Boot, recibe el texto

de la pregunta y lo envía a la API de OpenAI junto con un prompt predefinido, el cual permite generar una consulta SPARQL válida a partir del lenguaje natural introducido.

La consulta resultante se ejecuta sobre el motor Ontop, que actúa como puente entre la ontología y la base de datos Oracle. Ontop traduce automáticamente la SPARQL a SQL, accede a los datos reales y devuelve los resultados al backend, que los estructura y los envía al frontend para su presentación.

La información obtenida se muestra al usuario en el componente modal de ayuda de forma clara, ordenada y relacionada con la pregunta formulada. De este modo, el sistema permite ofrecer una ayuda contextualizada, pero guiada directamente por el propio usuario a través del lenguaje natural, sin necesidad de navegar por menús ni ayudas externas.

La implementación del código del botón de ayuda inteligente se detalla en el Anexo III, donde también se incluye el código fuente correspondiente. A continuación, en los siguientes apartados, se desglosan de forma explícita las distintas fases abordadas durante su desarrollo.

### **3.2.1 Arquitectura del backend en Spring Boot**

La arquitectura adoptada sigue una estructura modular y escalable, dividiendo claramente las responsabilidades de cada componente:

#### *Main.java*

Clase principal anotada con `@SpringBootApplication`. Se encarga de iniciar la aplicación con `SpringApplication.run()` y sirve como punto de entrada del sistema.

#### *AyudaController.java*

Define los endpoints REST del sistema. Gestiona todas las peticiones del usuario relacionadas con consultas en lenguaje natural (`/api/ayuda/consulta`). Esta clase centraliza el control del flujo de comunicación entre el frontend y los servicios de ayuda inteligente, sin necesidad de crear controladores separados por módulo funcional.

#### *OpenAiService.java*

Gestiona la interacción con la API de OpenAI. Construye el prompt, lanza la petición HTTP mediante `OkHttp` y devuelve la consulta SPARQL generada. Además, realiza una limpieza y validación de la consulta antes de su ejecución.

#### *SparqlService.java*

Ejecuta las consultas SPARQL generadas por OpenAI contra el motor Ontop, que

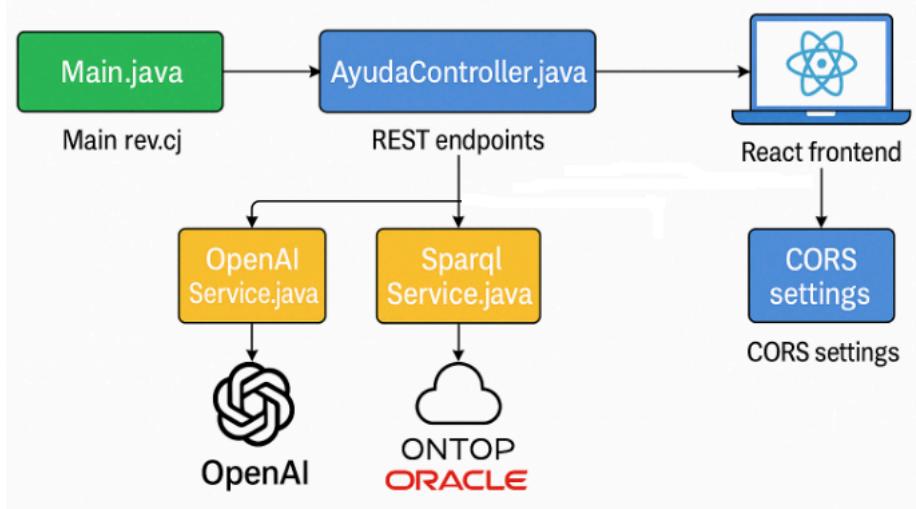
accede directamente a los datos de la base de datos Oracle. Procesa la respuesta JSON y la convierte a una estructura de listas y diccionarios legibles por el frontend.

#### *CorsConfig.java*

Configura los permisos CORS para permitir la comunicación entre el backend en localhost:8081 y el frontend desarrollado en React (normalmente en localhost:5173), especificando métodos, cabeceras y orígenes permitidos.

Imagen 3.2. Arquitectura del backend implementada en Spring Boot para el sistema de ayuda inteligente. Elaboración propia.

### Arquitectura del backend en Spring Boot



#### 3.2.2.1 Comunicación con Ontop y Oracle

Una vez generada la consulta SPARQL a partir de la pregunta del usuario, esta se ejecuta sobre Ontop, un motor de mapeo semántico que actúa como middleware entre la ontología del dominio (expresada en OWL) y la base de datos relacional

Oracle. Ontop permite mantener separadas la lógica semántica y la estructura física de los datos, facilitando una integración flexible y reutilizable.

Para establecer esta comunicación, se utiliza un archivo de mapeo .obda, que contiene la correspondencia entre las clases y propiedades ontológicas y las tablas y columnas de la base de datos. Por ejemplo, el concepto ontológico :Persona se vincula con la tabla AC\_PERSONAS, mapeando sus atributos como :perNombre, :perIdentificacion, :perUnidad, etc., mediante sentencias SQL definidas explícitamente.

Ontop traduce automáticamente la consulta SPARQL generada en el sistema a una instrucción SQL equivalente, la ejecuta directamente sobre la base de datos Oracle y devuelve los resultados en formato RDF. El backend, posteriormente, procesa estos resultados para:

- Eliminar información irrelevante o duplicada.
- Reestructurar los datos en un formato más adecuado para el frontend.
- Devolver una respuesta final en formato JSON, lista para ser mostrada en la interfaz o exportada a Excel.

El archivo completo de mapeo .obda se incluye en el Anexo IV como referencia técnica para comprender cómo se ha realizado la vinculación entre los conceptos definidos en la ontología y las estructuras reales de la base de datos Oracle.

El motor Ontop se ha desplegado en un contenedor Docker que se conecta directamente con la base de datos Oracle, permitiendo la ejecución transparente de consultas SPARQL traducidas automáticamente a SQL.

Además, en este anexo se documenta el proceso de instalación y configuración de Ontop en el contenedor Docker, facilitando su reutilización e integración en otros entornos.

### **3.2.2.2 Metodología de diseño del prompt para la generación de consultas SPARQL**

El desarrollo de la funcionalidad que permite transformar preguntas en lenguaje natural en consultas SPARQL ejecutables se ha basado en una metodología específica de diseño de prompts. Esta estrategia tiene como objetivo guiar al modelo de lenguaje de OpenAI para que genere consultas SPARQL correctas, sintácticamente válidas y semánticamente adaptadas al vocabulario de la ontología del sistema. El uso

de esta metodología asegura que las respuestas generadas puedan ejecutarse directamente en Ontop, accediendo a los datos reales almacenados en la base de datos Oracle.

A continuación se describen las fases seguidas:

#### *Definición del propósito del prompt*

El objetivo principal del prompt es convertir preguntas formuladas en lenguaje natural en consultas SPARQL que puedan ser ejecutadas de forma directa sobre el motor Ontop. Esto permite obtener información estructurada del sistema a partir del conocimiento modelado en la ontología y accesible mediante los mapeos OBDA previamente definidos.

#### *Identificación del dominio y vocabulario autorizado*

Para evitar errores de vocabulario y garantizar que las consultas SPARQL hagan referencia exclusivamente a propiedades y clases existentes, se ha creado una lista con el conjunto de términos permitidos. Esta lista se ha extraído directamente del vocabulario presente en la ontología y en los mapeos OBDA, incluyendo propiedades como :perNombre, :perCodigo, :ayuNombre, :ayuEjemplo, :trsDescripcion, :trsTipo, entre otras.

#### *Creación de ejemplos representativos*

Se ha aplicado una estrategia de aprendizaje por analogía (few-shot learning), incluyendo varios ejemplos reales de consultas SPARQL correctas. Estos ejemplos abarcan diferentes tipos de entidades y contextos frecuentes, como personas, trámites o ayuda sobre campos. Cada ejemplo muestra una pregunta en lenguaje natural y su correspondiente transformación a SPARQL. Esto ayuda al modelo a aprender patrones válidos de generación.

#### *Redacción de instrucciones claras y restrictivas*

El prompt incluye instrucciones precisas para que el modelo:

- Devuelva únicamente la consulta SPARQL.
- Evite explicaciones adicionales.
- Utilice solo el vocabulario autorizado.

- Aplique correctamente los tipos de datos (xsd:string, xsd:date, etc.).
- Evite inventar propiedades o clases no presentes en la ontología.

Estas restricciones minimizan los errores y aumentan la consistencia de las respuestas.

#### *Validación de las respuestas generadas*

Se ha llevado a cabo una fase de validación manual, verificando que las consultas generadas por OpenAI sean:

- Ejecutables en Ontop sin errores de sintaxis.
- Correctamente formadas desde el punto de vista semántico.
- Coherentes con la pregunta original del usuario.

Esta validación se ha realizado utilizando preguntas reales que un usuario podría formular durante su interacción con el sistema.

#### *Iteración y mejora progresiva del prompt*

A partir de los resultados obtenidos en la fase de validación, se ha iterado el diseño del prompt, ajustando las instrucciones, refinando los ejemplos y corrigiendo errores detectados. Esta mejora continua ha permitido optimizar la calidad de las consultas generadas, asegurando que el sistema sea funcional y fiable.

Esta metodología ha sido fundamental para integrar con éxito el componente de lenguaje natural en el botón de ayuda inteligente. La referencia al *prompt* utilizado se incluye en el Anexo IV y está disponible en el repositorio de GitHub.

### **3.2.2 Implementación del frontend y visualización de resultados**

La información obtenida desde el backend se muestra al usuario dentro de un modal interactivo de ayuda, que forma parte del componente de frontend implementado en *React*. Este modal puede abrirse mediante un botón flotante y ofrece

una interfaz sencilla e intuitiva para que el usuario formule preguntas en lenguaje natural.

Una vez realizada la consulta, el sistema analiza el contenido y muestra la respuesta de forma clara y estructurada, adaptándose al tipo de información devuelta. Este diseño busca ofrecer una experiencia de ayuda contextualizada, proactiva y guiada por el propio usuario, sin necesidad de navegar por menús complejos ni consultar documentación adicional.

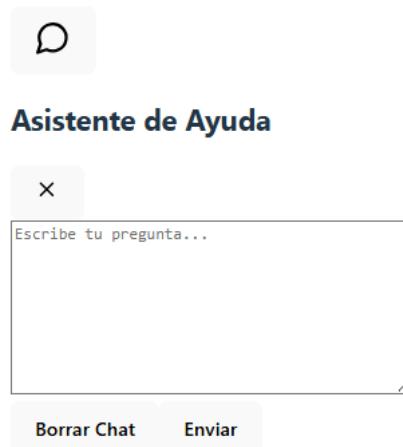
El usuario interactúa con el sistema a través de un botón de ayuda situado en la interfaz:

Imagen 3.3. Botón ayuda en la interfaz. Elaboración propia.



Al pulsarlo, se abre un componente *modal*, desde el cual puede introducir preguntas en lenguaje natural relacionadas con el ámbito “Atención al Ciudadano”:

Imagen 3.4. Interfaz tras pulsar sobre el botón ayuda modal. Elaboración propia.



La petición se envía desde el frontend al backend mediante una solicitud HTTP al endpoint `/api/ayuda/consulta`.

El diseño modular del botón permite su reutilización en distintas secciones sin necesidad de modificar su comportamiento interno.

La implementación completa del botón de ayuda inteligente, el código fuente en React, así como su explicación detallada, se incluye en el Anexo IV del presente documento.

### 3.2.3 Instalación en distintos entornos

El sistema ha sido diseñado con un enfoque modular y desacoplado para facilitar su despliegue en distintos entornos (desarrollo, pruebas, producción). A continuación se describen los aspectos clave que garantizan su portabilidad y facilidad de instalación:

- Backend en Spring Boot: La aplicación puede ejecutarse como un archivo JAR independiente. Todas las configuraciones específicas (URLs, credenciales, rutas de la ontología, endpoint SPARQL, etc.) se definen en el archivo `application.properties` o, alternativamente, mediante variables de entorno.
- Ontop y conexión con Oracle: El motor Ontop se ejecuta dentro de un contenedor Docker independiente, conectado a la base de datos Oracle a través de una red compartida. Los archivos de configuración (`.obda`, `.ttl`, y `.properties`) se montan en el contenedor durante su inicio.
- Configuración del endpoint SPARQL: La URL del endpoint expuesto por Ontop se configura en el backend como propiedad externa:
- Requisitos del sistema
  - Java 8
  - Docker instalado y configurado
  - Acceso a una instancia de Oracle con usuario autorizado para lectura
  - Sistema operativo compatible (Windows, Linux o macOS)
- Pasos de instalación
  - Adaptar los archivos `.obda` y `.ttl` a la estructura y datos del entorno Oracle.

- Iniciar el contenedor de Ontop con los archivos montados.
- Configurar el archivo *application.properties* del backend con las rutas y parámetros correspondientes.
- Ejecutar el backend de ayuda inteligente.
- Integrar el componente del botón de ayuda en el frontend en los puntos deseados de la aplicación.

Gracias a esta arquitectura desacoplada y parametrizable, el sistema puede instalarse fácilmente en distintos entornos sin necesidad de modificar el código fuente, garantizando su portabilidad, escalabilidad y facilidad de mantenimiento.

### **3.3 Evaluación y validación**

El objetivo de la evaluación es verificar que el botón de ayuda inteligente responde de forma útil, precisa y contextual a las preguntas del usuario, desde las más simples hasta las más complejas, basándose en la ontología y los datos reales de Oracle a través de SPARQL.

#### **3.3.1 Metodología de evaluación**

Con el fin de comprobar la eficacia del botón de ayuda inteligente implementado, se han definido varios casos de prueba organizados en diferentes iteraciones. Cada iteración incluye un conjunto de preguntas clasificadas en función de su complejidad: básicas, intermedias y avanzadas.

La evaluación ha sido realizada por la propia autora del sistema, con conocimiento profundo tanto del dominio como del funcionamiento interno de la solución desarrollada. Esto permite identificar de forma precisa la calidad de las respuestas ofrecidas y la cobertura funcional del sistema.

Durante las pruebas, las respuestas generadas por el sistema se analizan según los siguientes criterios:

- Relevancia: Se evalúa si la respuesta obtenida guarda relación directa con la pregunta planteada.
- Exactitud: Se comprueba si la información proporcionada es correcta, completa y coherente con los datos del sistema.

- Tiempo de respuesta: Se registra el tiempo que tarda el sistema en devolver una respuesta, considerando la experiencia del usuario.
- Reformulación: Se anota si la pregunta inicial no ha sido comprendida correctamente por el sistema, y ha sido necesario reformularla para obtener una respuesta adecuada.
- Nivel dificultad: Se clasifica cada pregunta como básica, intermedia o avanzada, en función de su complejidad.
- Sección evaluada: Se especifica el módulo funcional o ámbito temático al que corresponde la consulta (ej. Personas, Trámites, Empleo Público...).
- Observaciones: Se recogen notas adicionales sobre el comportamiento del sistema o incidencias detectadas.

Este enfoque metodológico, aunque limitado a una evaluación individual, permite detectar tanto los puntos fuertes como las posibles limitaciones del sistema, sirviendo como base para realizar ajustes y mejoras en iteraciones sucesivas.

### 3.3.2 Resultados de la evaluación

A lo largo del proceso de validación, se han realizado diversas iteraciones de prueba para evaluar el comportamiento del sistema en distintos escenarios. En cada iteración se han planteado preguntas de diferente nivel de complejidad, desde cuestiones básicas sobre campos individuales hasta consultas más avanzadas que implican múltiples entidades o condiciones.

Los resultados se han registrado en tablas que reflejan, para cada pregunta, la respuesta obtenida, su relevancia, la exactitud de la información proporcionada, el tiempo de respuesta, y si fue necesario reformular la consulta. Este enfoque progresivo permite observar la evolución del sistema y detectar patrones de funcionamiento que orientan futuras mejoras.

Dado el volumen de información evaluada, los resultados completos se han incluido en el *Anexo VI Evaluación del Botón de Ayuda Inteligente*, donde se detallan todos los casos de uso analizados. A continuación, se ofrece un resumen general con las principales conclusiones extraídas de dichas iteraciones:

Métrica	Resultado
Total de preguntas evaluadas	32
Porcentaje de respuestas relevantes	75%
Porcentaje de respuestas correctas	75%
Tiempo medio de respuesta	0.24

### 3.3.3 Análisis de resultados

Tras realizar una evaluación sistemática de 32 preguntas distribuidas entre los distintos módulos funcionales del sistema (Personas, Registros, Entidad Local, Departamentos, Trámites y Servicios, Fases del Procedimiento y subfases, Empleo Público y etapas, Criterios de Búsqueda y Operaciones), se ha podido medir de forma cuantitativa y cualitativa el rendimiento de la ayuda inteligente basada en ontología.

El 75 % de las preguntas obtuvieron respuestas relevantes, lo que demuestra una adecuada cobertura semántica en la ontología y el sistema de ayuda. Las respuestas fueron especialmente precisas en los módulos más estructurados como *Personas, Trámites y Servicios* y *Fase de Instrucción*. Sin embargo, se observaron carencias en campos o validaciones no contempladas en la tabla AC\_AYUDA\_CAMPO o en los criterios de búsqueda, lo que representa una oportunidad de mejora futura.

Del total de preguntas relevantes, la exactitud también fue del 75 %. Esto indica que la mayoría de las respuestas no solo se ajustan al contexto semántico, sino que proporcionan información comprensible, ejemplos concretos y reglas asociadas. Las respuestas de nivel medio y avanzado mostraron una mayor necesidad de precisión semántica y, en algunos casos, dependieron del uso correcto del diccionario de campos y del entrenamiento del modelo.

El tiempo medio de respuesta fue de 0.24 segundos, lo que demuestra una excelente capacidad del sistema para ejecutar consultas SPARQL sobre Ontop en tiempo real, ofreciendo una experiencia fluida para el usuario final. Todas las respuestas se generaron por debajo del umbral de 3 segundos, considerado óptimo en aplicaciones interactivas. A tener en cuenta que disminuye considerablemente el tiempo de respuesta en 0.02 segundos si ésta se vuelve a enviar por segunda vez y sucesivas.

Solo el 9 % de las preguntas (3 de 32) necesitaron una reformulación para ser comprendidas por el sistema. Estas situaciones estuvieron relacionadas con ambigüedades semánticas (ej. *fase fin* o *normativa*) o con la ausencia de contexto suficiente en el prompt. La baja tasa de reformulación necesaria sugiere que el sistema es capaz de interpretar correctamente preguntas formuladas en lenguaje

natural, incluso cuando no coinciden exactamente con los nombres técnicos de los campos.

El uso de ejemplos y formatos esperados en las respuestas dan más coherencia y comprensión para el usuario. Se detectaron campos que no están presentes en la tabla de ayuda, lo que limita la capacidad del sistema para ofrecer orientación. A tener en cuenta que, algunas preguntas de lógica avanzada realizadas (operaciones sobre entidades no creadas, validaciones cruzadas, etc.) no son actualmente abordables sin razonamiento adicional.

A partir de los resultados obtenidos, se proponen las siguientes acciones para incrementar la efectividad, cobertura y utilidad del asistente de ayuda inteligente:

1. Ampliación de la base de conocimientos semántica:

Incluir en la tabla AC\_AYUDA\_CAMPO nuevos campos aún no contemplados (como validaciones de fechas, casuísticas excepcionales o diferencias entre campos relacionados como teléfono interno/externo), especialmente en módulos como *Registros*, *Entidad Local* y *Etapa de Empleo Público*.

2. Mejora del prompt contextual de OpenAI:

Incorporar ejemplos específicos por módulo y reforzar el diccionario técnico-natural para reducir aún más la necesidad de reformulación. Esto permitiría entender mejor preguntas imprecisas o genéricas con mayor acierto.

3. Soporte a validaciones lógicas complejas (futuro):

Explorar una capa de razonamiento adicional que permita responder preguntas que impliquen reglas condicionales o inferencias (ej. "¿puedo modificar una entidad que no ha sido creada?"). Esto puede lograrse mediante reglas SWRL o integración futura con modelos de LLM más avanzados. Como ampliación futura, se podría incorporar razonamiento automático mediante motores como Pellet y reglas SWRL para validar operaciones no permitidas o inferir estados del sistema de forma semántica.

4. Entrenamiento específico por módulo (si se usa IA):

Permitir que el sistema entrene respuestas frecuentes por sección funcional, adaptando el comportamiento a cada dominio (por ejemplo, *Trámites*, *Personas*, *Fases*, etc.).

5. Mejora de la interfaz de respuesta para explicaciones complejas:

Añadir pestañas o bloques plegables que agrupen "definición", "ejemplo",

"reglas" y "uso", permitiendo una mejor legibilidad en preguntas de nivel medio o avanzado.

## Parte 4

### 4. Desarrollo de la Detección de errores

El objetivo es identificar inconsistencias o configuraciones incorrectas en los datos del sistema, que puedan afectar a la calidad de la información ofrecida y a la eficiencia de los procesos administrativos.

A diferencia de la concepción inicial centrada en errores técnicos derivados de cambios de versión, esta funcionalidad se ha reorientado hacia un enfoque semántico, más alineado con el modelo ontológico del sistema. En lugar de analizar logs o fallos

técnicos, se abordan errores conceptuales detectables a partir de reglas o condiciones sobre los datos almacenados.

Para ello, se reutiliza la misma arquitectura implementada en el *apartado 2.2*: se parte de una ontología que modela el dominio de atención al ciudadano, conectada mediante mapeos OBDA al sistema de gestión de bases de datos Oracle. Sobre esta base se ejecutan consultas SPARQL capaces de detectar registros incompletos, relaciones ausentes o inconsistencias lógicas, como trámites sin unidad gestora, fechas de finalización anteriores a fechas de inicio o personas sin departamentos de adscripción.

El resultado es una solución que permite identificar errores semánticos de forma estructurada, trazable y basada en el modelo de conocimiento, facilitando su análisis y resolución.

Además, los errores detectados se representan como instancias OWL dentro de la ontología mediante el sistema de mapeo, permitiendo su integración futura en mecanismos de notificación, visualización o generación de alertas proactivas.

## 4.1 Tipología de errores considerados

La detección automática de errores se centra en identificar incoherencias y omisiones en los datos que puedan comprometer la fiabilidad del sistema o impedir el correcto desarrollo de los procedimientos administrativos.

Estos errores no son técnicos (como fallos de ejecución o errores de infraestructura), sino semánticos, ya que afectan a la lógica del dominio y a las relaciones entre entidades.

A continuación, se presentan los tipos de errores considerados más relevantes en el contexto del sistema de atención al ciudadano:

Tipo de error	Descripción	Ejemplo
Error de asignación	Se produce cuando una entidad no está asociada a otra obligatoria según la lógica del dominio.	Un trámite sin departamento asignado.
Error de relación ausente	Falta una relación entre entidades que debería existir según las reglas del sistema.	Una persona no tiene asignado un departamento.

Error de consistencia temporal	Las fechas registradas no siguen un orden lógico.	Fecha de finalización anterior a la de inicio.
Error de datos faltantes	Faltan valores obligatorios en campos esenciales.	Un procedimiento sin código identificador o sin descripción.
Error de tipo o formato	El valor registrado no cumple el formato esperado.	Un campo numérico contiene texto no interpretable.
Error de duplicidad	Se produce cuando dos o más registros representan la misma entidad de manera redundante o inconsistente.	Dos personas registradas con el mismo DNI pero diferentes identificadores.

Cada uno de estos errores puede ser identificado mediante consultas SPARQL ejecutadas sobre los datos reales de Oracle, gracias al mapeo semántico realizado con Ontop.

Una vez detectados, los errores se pueden registrar como instancias OWL dentro de la ontología, permitiendo su trazabilidad, análisis y posible visualización futura.

Aunque la aplicación web implementa validaciones técnicas tradicionales en la entrada de datos, el sistema de detección semántica propuesto en este proyecto proporciona una capa adicional de validación posterior, enfocada en la integridad y consistencia global de los datos almacenados. Esta verificación semántica permite identificar errores que pueden haber escapado a las validaciones iniciales, surgido por cargas masivas, modificaciones posteriores o cambios en las reglas de negocio.

## 4.2 Representación semántica de errores en la ontología

Con el fin de integrar los errores detectados en el modelo semántico del sistema, se ha ampliado la ontología con una nueva clase *:Error*, que permite representar de forma estructurada las incidencias identificadas mediante consultas SPARQL.

Esta clase se ha diseñado para ser genérica y extensible, de manera que cualquier error relacionado con la integridad de los datos pueda ser modelado como una instancia OWL. Además, se han definido propiedades específicas que permiten describir el error, indicar la entidad afectada y registrar metadatos como la fecha de detección.

### Clases y propiedades añadidas

- :Error → Clase principal que representa un error detectado.
- :ErrorConsistencia → Subclase para errores tipo: asignación, relación ausente, consistencia temporal y datos faltantes.
- :ErrorFormato → Subclase para errores tipo o formato en los datos.
- :ErrorDuplicidad → Subclase para errores datos duplicados.
- :errorDescripcion → Propiedad de tipo literal que describe la causa o naturaleza del error.
- :errorFechaDetec → Propiedad de tipo fecha que indica cuándo fue detectado.
- :errorDetectadoEn → Propiedad de tipo objeto que vincula el error con la entidad afectada (ej. un trámite, una persona...).

Tras la validación de la ontología base realizada en el *apartado 2.1.4*, y en el marco de las necesidades específicas de la detección de errores semánticos, se ha procedido a una ampliación controlada del modelo ontológico. Esta extensión no altera la estructura ni la coherencia del modelo original validado, sino que introduce nuevas clases y propiedades orientadas exclusivamente a la representación estructurada de incidencias de calidad de datos. Esta ampliación mantiene los principios de modularidad y extensibilidad definidos en el diseño inicial.

## 4.3 Evaluación y validación

Para evaluar la efectividad del sistema de detección automática de errores, se han ejecutado diversas consultas SPARQL sobre el motor Ontop, aplicadas a datos ficticios del sistema. Estas consultas han sido diseñadas para identificar inconsistencias frecuentes en las entidades modeladas, tales como relaciones obligatorias no cumplidas, incoherencias temporales o ausencia de información esencial.

### 4.3.1 Metodología de evaluación

La evaluación de la funcionalidad de detección automática de errores se ha planteado siguiendo un enfoque progresivo y estructurado, centrado en comprobar la capacidad del sistema para identificar inconsistencias conceptuales en los datos ficticios del sistema.

Para ello, se han diseñado varias consultas SPARQL ejecutadas sobre Ontop, conectadas a la base de datos Oracle mediante el archivo de mapeo .obda. Estas consultas permiten detectar errores como trámites sin unidad gestora, fechas mal configuradas o registros sin datos esenciales.

Cada consulta se ha evaluado en función de los siguientes criterios:

- Relevancia: Se analiza si los errores detectados responden a condiciones lógicas o semánticas que realmente deberían evitarse en el sistema.
- Exactitud: Se comprueba si los resultados devueltos por la consulta son correctos y se corresponden con situaciones erróneas reales.
- Tiempo de ejecución: Se registra el tiempo medio que tarda cada consulta en completarse sobre los datos reales.
- Cobertura: Se observa cuántos errores son detectados y en qué módulos del sistema.
- Trazabilidad semántica: Se evalúa si los errores se han representado correctamente como instancias de la clase *:Error* en la ontología.
- Reutilización de consultas: Se valora si las consultas pueden integrarse en procesos futuros de validación o mantenimiento.

Esta metodología permite no solo verificar el correcto funcionamiento de las consultas, sino también valorar su aplicabilidad en entornos reales y su potencial como parte de un sistema de supervisión semántico.

#### 4.3.2 Resultados de la evaluación

Los casos de uso detallados y los errores detectados se encuentran documentados en el *Anexo VII: Evaluación “Detección de errores”*, donde se analizan individualmente los resultados de cada consulta aplicada.

A continuación, se presenta un resumen de los resultados obtenidos:

Métrica	Resultado
Total de casos evaluados	6
Total errores detectados	9 (según los códigos exactos mostrados)
Tipos de errores identificados	6 (asignación, relación ausente, consistencia temporal, datos faltantes, formato, duplicidad)
Tiempo medio de respuesta	0.26 s (promedio de todos los tiempos)
Porcentaje errores ok mapeados OWL	100 % (todos los errores tienen trazabilidad semántica con clase <i>:Error</i> )

#### **4.3.3 Análisis de resultados**

Los resultados obtenidos reflejan que el sistema propuesto permite detectar con éxito distintos tipos de errores semánticos en los datos administrativos, tales como asignaciones incorrectas, relaciones ausentes, inconsistencias temporales, campos obligatorios vacíos, errores de formato y duplicidades. Cada uno de estos errores ha sido representado como una instancia de la clase :Error o de sus subclases especializadas (:ErrorConsistencia, :ErrorFormato, :ErrorDuplicidad), asegurando una trazabilidad clara y estructurada de los fallos detectados.

La ejecución de las consultas SPARQL sobre Ontop ha demostrado ser rápida y reutilizable, con tiempos de respuesta inferiores a un segundo en todos los casos. Esto permite su potencial integración en procesos de validación periódica, auditoría o visualización interactiva. Asimismo, el uso de una ontología de dominio bien definida facilita la extensión del sistema con nuevas reglas o categorías de errores sin alterar su arquitectura.

Sin embargo, existen limitaciones importantes que deben señalarse. La evaluación se ha restringido a seis casos representativos, seleccionados por su relevancia semántica y viabilidad técnica, lo que no garantiza una cobertura completa de todos los posibles errores existentes en el sistema. Además, no se ha implementado un sistema de visualización de errores integrado, ni mecanismos automáticos de reparación o notificación. Por tanto, el sistema actual debe considerarse como una *prueba de concepto funcional*, con potencial para escalar y evolucionar en futuras fases de desarrollo.

## **Parte 5**

### **5. Desarrollo de Consultas personalizadas**

El objetivo de esta funcionalidad es permitir al usuario realizar consultas personalizadas sobre los datos del sistema, accediendo a información estructurada y actualizada directamente desde la base de datos relacional. A diferencia de la ayuda inteligente, centrada en ofrecer explicaciones contextuales, esta funcionalidad está orientada a la generación de listados completos que pueden ser exportados en formato Excel para su posterior análisis o utilización en tareas administrativas.

La funcionalidad de consultas personalizadas constituye una extensión natural del sistema de ayuda inteligente descrito en la Parte 3. Ambas comparten una arquitectura técnica común, basada en el uso de una ontología del dominio, el motor SPARQL OnTop conectado a una base de datos relacional Oracle, y un backend desarrollado en Spring Boot que orquesta las peticiones generadas desde el frontend.

En esta sección se documenta la implementación específica de este módulo, incluyendo los casos de uso más representativos, el proceso de generación y ejecución de las consultas SPARQL, y el sistema de exportación de resultados integrado en la interfaz del usuario.

## 5.1 Adaptación técnica específica para la generación de listados

El sistema reutiliza los componentes tecnológicos ya descritos en la Parte 3 del presente trabajo: la ontología definida en OWL, el motor de mapeo OnTop conectado a Oracle, el backend desarrollado en Spring Boot y la generación de consultas SPARQL a partir de lenguaje natural mediante la API de OpenAI

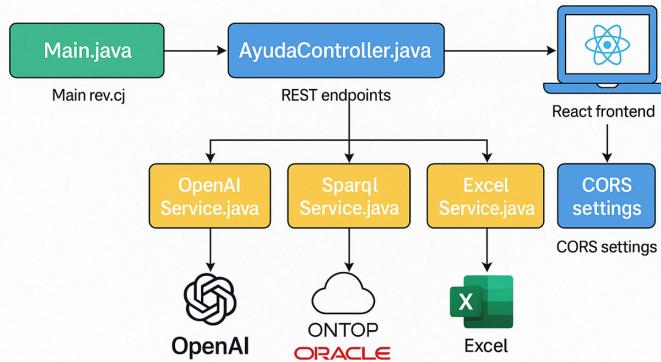
Para permitir la exportación de resultados en formato Excel, se ha añadido un nuevo servicio denominado *ExcelService.java*, que procesa los datos devueltos por OnTop y los transforma en un archivo descargable. Este componente se activa cuando la respuesta generada por el asistente contiene un listado de elementos estructurados, como personas, trámites o empleo público, que pueden ser útiles para su análisis externo o impresión.

A nivel técnico, la exportación se gestiona mediante la librería *Apache POI*, que permite la creación dinámica de hojas de cálculo en formato .xlsx. Este servicio se encuentra desacoplado del resto de componentes para favorecer su reutilización en otros módulos.

La arquitectura resultante se muestra a continuación, incorporando este nuevo servicio a la estructura modular del backend:

Imagen 5.1. Arquitectura del backend implementada en Spring Boot para el sistema de ayuda inteligente, incluyendo exportación de datos en Excel.  
Elaboración propia.

## Arquitectura del backend en Spring Boot



En el Anexo V se incluye el código fuente completo de este servicio, así como su integración técnica dentro del sistema. También se referencia el repositorio de GitHub donde puede consultarse y descargarse.

## 5.2 Evaluación y validación

El objetivo de esta evaluación es comprobar si el sistema es capaz de interpretar correctamente las peticiones de consultas formuladas por el usuario en lenguaje natural, generar automáticamente las consultas SPARQL correspondientes, y devolver los resultados basados en los datos reales almacenados en la base de datos Oracle, con la posibilidad de exportarlos en formato Excel.

### 5.2.1 Metodología de evaluación

Se han diseñado distintas consultas simuladas, clasificado según su nivel de dificultad (básica, intermedia y avanzada) y se han analizado en función de los siguientes criterios:

- Relevancia: Se evalúa si los datos obtenidos responden de manera útil y pertinente a la consulta.
- Exactitud: Se comprueba que los resultados coincidan con los datos reales existentes en la base de datos.
- Tiempo de respuesta: Se mide el tiempo que tarda el sistema en procesar la consulta y mostrar los resultados.
- Formato del resultado: Se observa si los datos se presentan de forma comprensible y permiten su exportación (a Excel).
- Reformulación: Se anota si la pregunta inicial fue mal interpretada y tuvo que reformularse para obtener una respuesta adecuada.

- Nivel dificultad: Se clasifica cada consulta como básica, intermedia o avanzada, en función de su complejidad.
- Sección evaluada: Se indica el módulo funcional (Personas, Trámites, Empleo Público, etc.) al que corresponde la consulta.
- Observaciones: Se recogen notas adicionales sobre el comportamiento del sistema.

Este enfoque metodológico permite identificar el grado de comprensión semántica del sistema, así como su capacidad para generar resultados útiles y directamente aplicables en el contexto de atención al ciudadano.

### 5.2.2 Resultados de la evaluación

Durante el proceso de validación de la funcionalidad de consultas personalizadas en lenguaje natural, se han llevado a cabo distintas iteraciones de prueba basadas en consultas de uso representativos. Cada consulta simula una necesidad dentro del contexto del sistema de atención al ciudadano, y pone a prueba la capacidad del sistema para interpretar correctamente la solicitud, generar una consulta SPARQL adecuada y devolver resultados útiles, exactos y exportables.

Las consultas han sido creadas y ejecutadas por la autora del sistema, clasificándolos según su nivel de complejidad (básico, intermedio y avanzado). Se ha evaluado el comportamiento del sistema atendiendo a la fidelidad de la interpretación semántica, la calidad de los resultados obtenidos, el tiempo de respuesta, la presentación de los datos y la necesidad de reformulación en aquellos casos donde el sistema no comprendía correctamente la petición original.

Dado el nivel de detalle y volumen de información evaluada, los resultados completos se incluyen en el *Anexo VIII: Evaluación de las Consultas Personalizadas*, donde se documenta cada consulta generada, la valoración obtenida y observaciones relevantes.

A continuación, se presenta un resumen general con las principales conclusiones extraídas de las iteraciones realizadas:

Métrica	Resultado
Total de consultas evaluadas	14
Porcentaje de respuestas relevantes	100%
Porcentaje de respuestas correctas	92.8%

Tiempo medio de respuesta	~1.87 s 1 <sup>a</sup> iterac. ~0.027 s 2 <sup>o</sup> iterac.
Reformulaciones necesarias	6
Porcentaje de resultados correctamente exportados	100%

### 5.2.3 Análisis de resultados

Las consultas de nivel básico y medio fueron resueltas correctamente en todos los casos, sin necesidad de reformulación ni ajustes. En el nivel avanzado, la tasa de éxito también fue alta, pero la mayoría de las preguntas requirió alguna intervención manual o reformulación del prompt, sobre todo cuando implicaban relaciones complejas entre múltiples entidades.

## Parte 6

### 6. Conclusiones

Ver en memoria final.

## 6.1 Líneas de trabajo futuro

Ver en memoria final.

# ANEXOS

## Anexo I. Ontología “Atención al Ciudadano” (ontoAC)

Este anexo recoge una serie de capturas de pantalla obtenidas desde Protégè que ilustran la ontología desarrollada en el presente trabajo:

**Metrics**

Axiom	1.106
Logical axiom count	478
Declaration axioms count	218
Class count	16
Object property count	23
Data property count	166
Individual count	12
Annotation Property count	2

**Class axioms**

SubClassOf	8
EquivalentClasses	0
DisjointClasses	2
GCI count	0
Hidden GCI Count	0

**Object property axioms**

SubObjectPropertyOf	0
EquivalentObjectProperties	0
InverseObjectProperties	0
DisjointObjectProperties	0
FunctionalObjectProperty	0
InverseFunctionalObjectProperty	0
TransitiveObjectProperty	0
SymmetricObjectProperty	0
AsymmetricObjectProperty	0
ReflexiveObjectProperty	0
IrreflexiveObjectProperty	0
ObjectPropertyDomain	23
ObjectPropertyRange	23
SubPropertyChainOf	0

**Data property axioms**

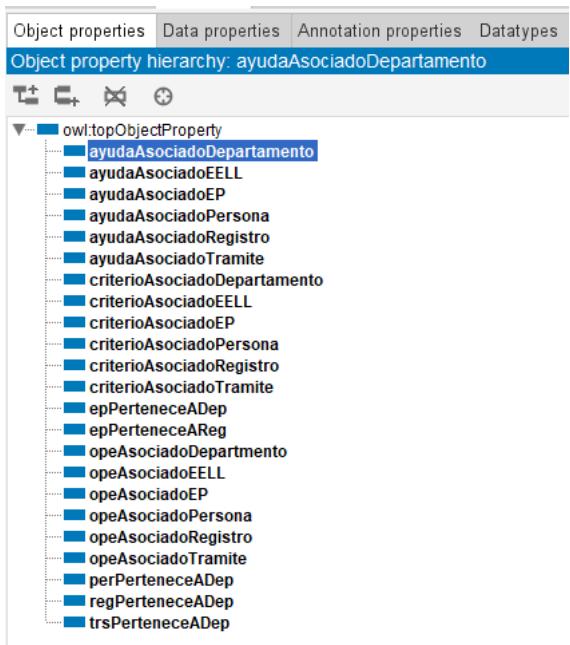
SubDataPropertyOf	0
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	0
DataPropertyDomain	166
DataPropertyRange	166

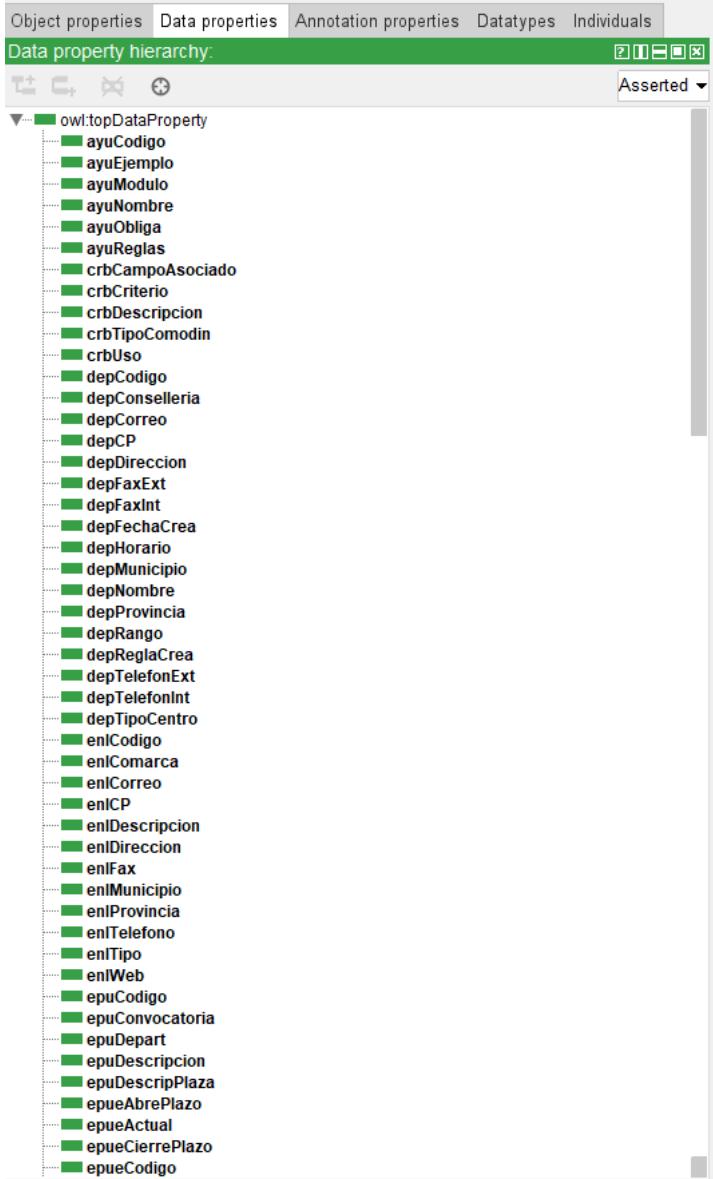
**Individual axioms**

ClassAssertion	11
ObjectPropertyAssertion	6
DataPropertyAssertion	70
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0
DifferentIndividuals	0

**Annotation axioms**

AnnotationAssertion	410
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0





ontology (<http://example.org/ontologia>) : [C:\workspace\TFM-dyadic-inteligente\ontoAC\ontoAC.owl]

File Edit View Reasoner Tools Refactor Window Ontop Help

< > [ontologia \(<http://example.org/ontologia>\)](#)

active ontology Entities Classes Individuals by class DL Query OntoGraf Ontop Mappings SWRLTab Debugger SPARQL Query

Queries Start Entitled Test Cases Non-Entitled Test Cases

Yess Start to check the consistency and coherency of the ontology.

When the ontology is inconsistent and/or incoherent, queries in the form of axioms can be answered here in order to:

Coherent (& Consistent) Ontology!

The ontology "ontologia (<http://example.org/ontologia>)" is coherent and consistent

Aceptar

Possible Faulty Axioms

Enter a string to search within the Possible Faulty Axioms

Case sensitive Whole words Ignore white space Regular expression

Class axioms Object properties Data properties Individuals

1-100 of 478 axioms

possible Ontology Repairs

Entitled Test Cases Non-Entitled Test Cases

Saved Test Cases

0 axioms

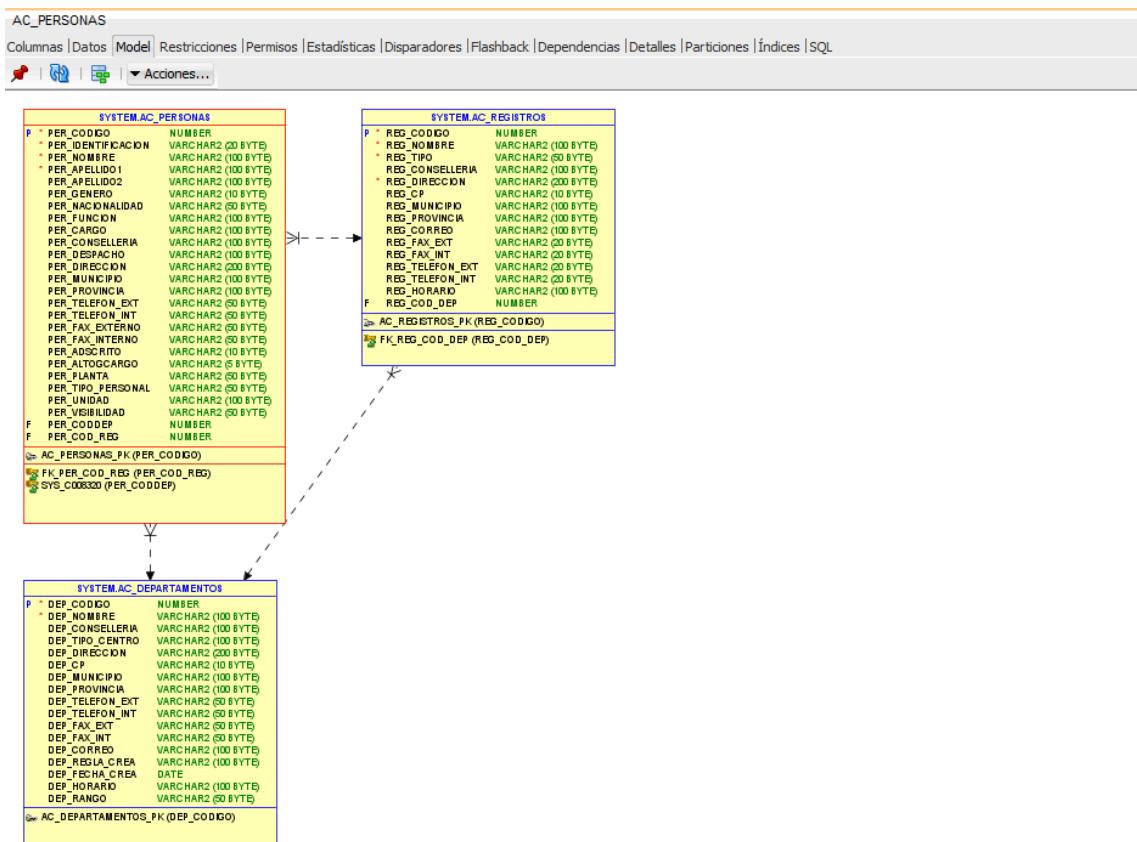
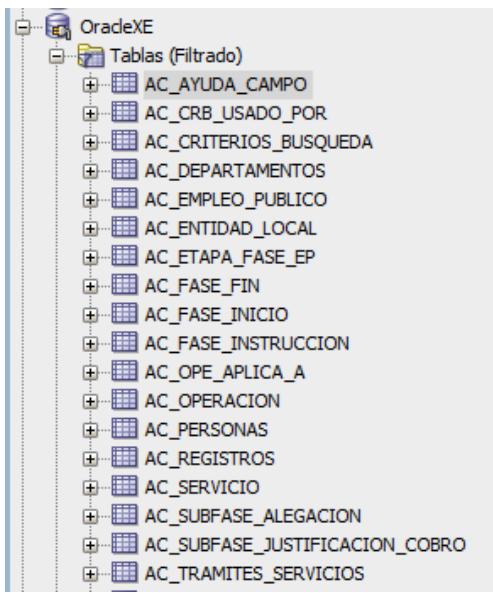
To use the reasoner click Reasoner > Start reasoner Show Inferences

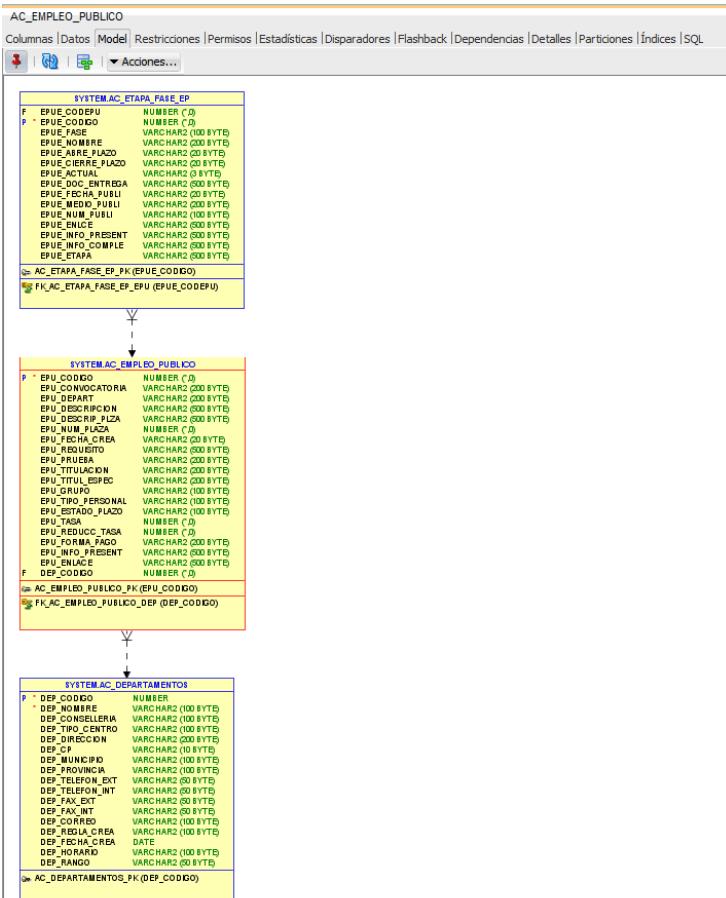
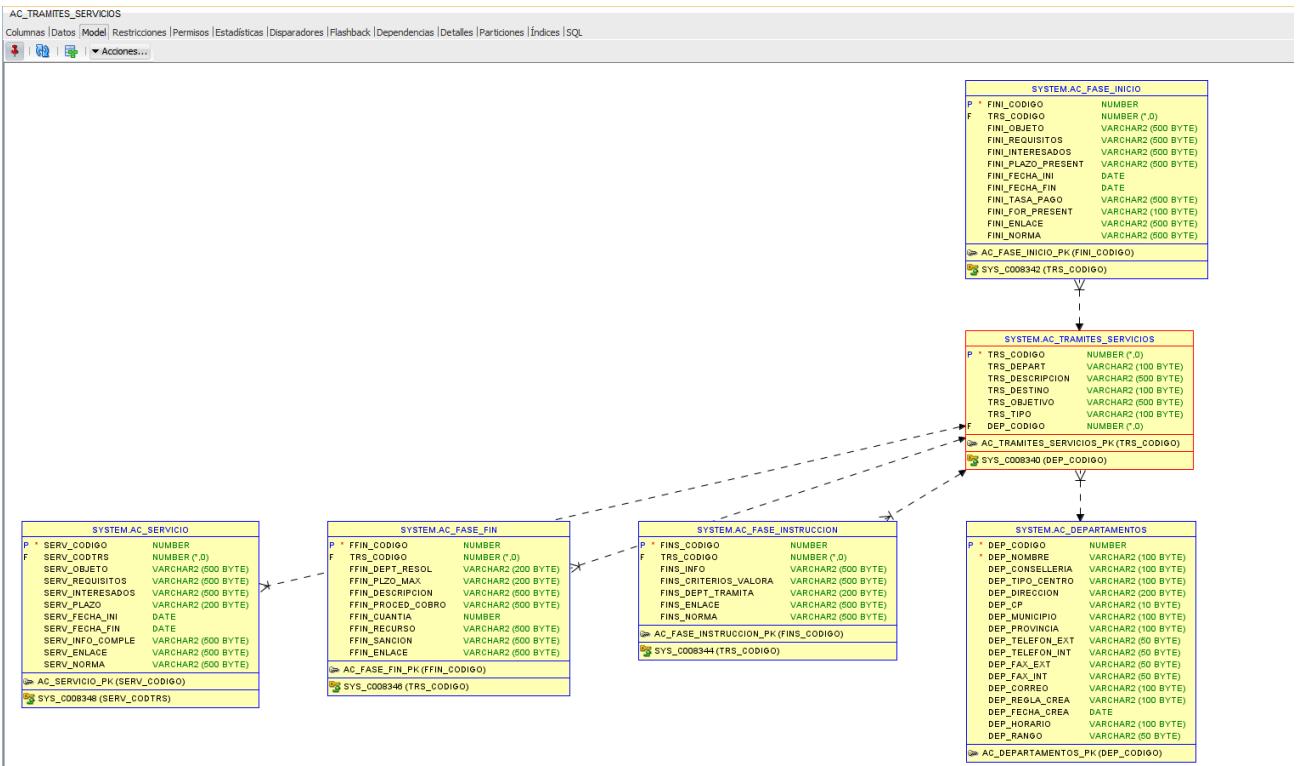
La ontología completa puede consultarse en formato OWL, RDF y Turtle en el repositorio del proyecto:

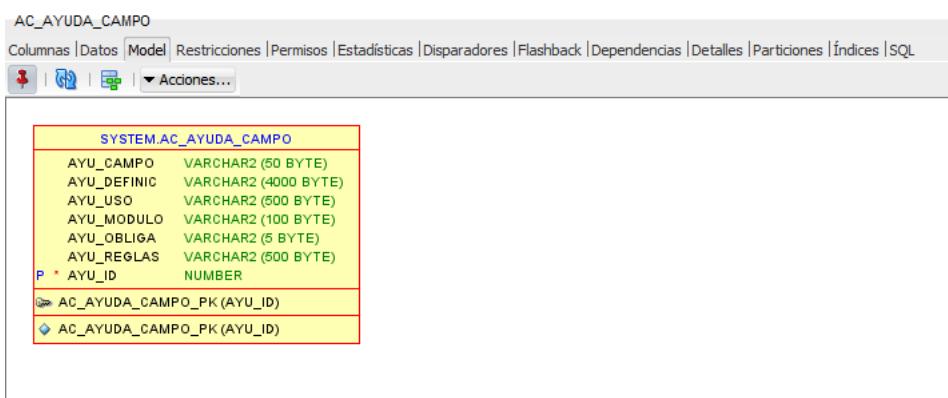
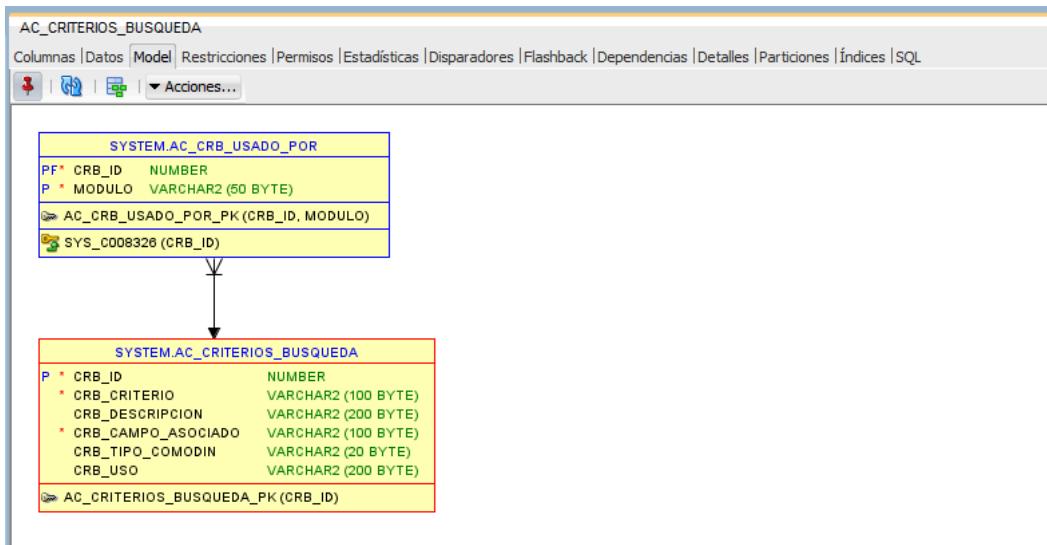
<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/ontoAC>

## Anexo II. Modelo base de datos AC

Este anexo recoge una serie de capturas de pantalla obtenidas desde Oracle SQL Developer que ilustran el modelo de base de datos en el presente trabajo:





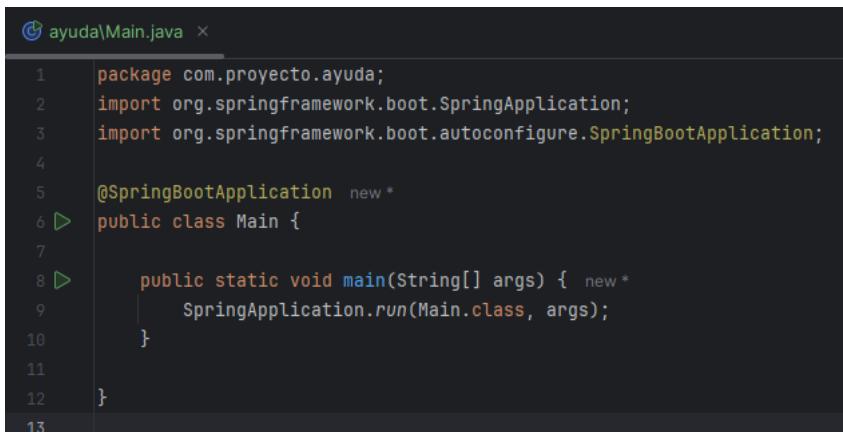


## Anexo III. Doc. Backend y frontend “Botón Ayuda Inteligente”

Este anexo recoge el análisis detallado del backend y frontend desarrollados para la implementación del sistema de ayuda inteligente en el proyecto. A continuación, se describen los principales componentes que forman parte de la arquitectura.

### Documentación del Backend (Spring Boot)

#### A.1 Main.java (Clase principal de arranque de Spring Boot)



```
ayuda>Main.java
1 package com.proyecto.ayuda;
2 import org.springframework.boot.SpringApplication;
3 import org.springframework.boot.autoconfigure.SpringBootApplication;
4
5 @SpringBootApplication new *
6 public class Main {
7
8     public static void main(String[] args) { new *
9         SpringApplication.run(Main.class, args);
10    }
11
12 }
13
```

El archivo Main.java contiene la clase Main, que constituye el punto de entrada principal de la aplicación. La anotación `@SpringBootApplication` habilita la configuración automática de Spring, el escaneo de componentes (`@ComponentScan`) y la carga del contexto de aplicación.

Este archivo ejecuta el método main, el cual lanza la aplicación Spring Boot con un servidor embebido (por defecto, Tomcat). Desde aquí se inicializan todos los controladores REST, servicios y configuraciones necesarias para el funcionamiento del sistema.

En este caso, no se incluye lógica de inicialización personalizada en el main, ya que la carga de los recursos necesarios (ontología, mapeos OBDA y configuración SPARQL) se realiza de forma automática al iniciar Ontop y al invocar los servicios REST.

Este diseño minimalista facilita la reutilización del backend en otros entornos y mejora su mantenibilidad.

## A.2 AyudaController.java (Controlador de ayuda en lenguaje natural)

```
© AyudaController.java ×
19  public class AyudaController {
31
32      /**
33      * Endpoint principal para procesar consultas en lenguaje natural.
34      * Genera la consulta SPARQL con ayuda de OpenAI y la ejecuta sobre Ontop (base de datos relacional).
35      * @param pregunta Pregunta en lenguaje natural formulada por el usuario.
36      * @return Respuesta estructurada con la consulta SPARQL generada y los resultados obtenidos desde Oracle.
37      */
38      @GetMapping("/consulta") no usages new
39      public Map<String, Object> obtenerConsulta(@RequestParam String pregunta) {
40          Map<String, Object> respuesta = new HashMap<>();
41
42          try {
43              System.out.println("Pregunta recibida del usuario: " + pregunta);
44
45              // 1. OpenAI genera la consulta SPARQL
46              String consultaSPARQL = openAiService.generarConsultaSPARQL(pregunta);
47              String consultaLimpia = openAiService.limpiarConsulta(consultaSPARQL);
48
49              System.out.println("Consulta SPARQL generada por OpenAI:\n" + consultaLimpia);
50
51              // 2. Validación: comprueba si la respuesta realmente es una consulta SELECT
52              if (!consultaLimpia.toUpperCase().contains("SELECT")) {
53                  System.err.println("⚠ Error: la respuesta de OpenAI no contiene una consulta SELECT válida.");
54                  respuesta.put("error", "OpenAI no generó una consulta SPARQL válida.");
55                  return respuesta;
56              }
57
58              // 3. Ejecuta la consulta SPARQL en Ontop
59              System.out.println("Ejecutando consulta en Ontop (Oracle + Ontología)");
60              Map<String, Object> resultado = sparqlService.ejecutarConsultaOntop(consultaLimpia);
61
62              // 4. Devuelve respuesta estructurada
63              respuesta.put("consultaSPARQL", consultaLimpia);
64              respuesta.put("resultado", resultado.get("resultado"));
65
66              System.out.println("Consulta ejecutada correctamente y datos devueltos.");
67              return respuesta;
68      }
}
```

AyudaController expone el endpoint REST encargado de recibir las preguntas formuladas por los usuarios en lenguaje natural y procesarlas mediante técnicas de procesamiento del lenguaje y Web Semántica.

El método obtenerConsulta invoca a OpenAI para generar una consulta SPARQL, que posteriormente se ejecuta sobre Ontop. Los resultados se devuelven en formato estructurado JSON para su visualización en el frontend.

La clase está anotada con `@RestController`, lo cual indica que se trata de un componente de Spring encargado de gestionar solicitudes HTTP y devolver

directamente respuestas JSON. Esta anotación es una combinación de `@Controller` y `@ResponseBody`, y permite construir APIs RESTful de manera sencilla.

### A.3 OpenAiService.java (Servicio de integración con OpenAI)

```
⑤ OpenAiService.java ×
19  public class OpenAiService {
41      private final Map<String, String> cacheSPARQL = Collections.synchronizedMap(new LinkedHashMap<String, String>
42          @Override 2 usages new *
43          @† protected boolean removeEldestEntry(Map.Entry<String, String> eldest) {
44              return size() > cacheSize;
45          }
46      );
47
48      private final OkHttpClient client = new OkHttpClient.Builder() 1 usage
49          .callTimeout(Duration.ofSeconds(10))
50          .connectTimeout(Duration.ofSeconds(5))
51          .readTimeout(Duration.ofSeconds(10))
52          .build();
53
54      public String generarConsultaSPARQL(String pregunta) throws IOException { 3 usages new *
55          if (cacheSPARQL.containsKey(pregunta)) {
56              System.out.println("\u26aa Usando caché para la pregunta: " + pregunta);
57              return cacheSPARQL.get(pregunta);
58          }
59
60          String prompt = construirPrompt(pregunta);
61          String jsonBody = generarJsonRequestDesdePrompt(prompt);
62
63          RequestBody body = RequestBody.create(jsonBody, MediaType.parse("application/json"));
64          Request request = new Request.Builder()
65              .url(apiUrl)
66              .header(name: "Authorization", value: "Bearer " + apiKey)
67              .header(name: "Content-Type", value: "application/json")
68              .post(body)
69              .build();
70
71          try (Response response = client.newCall(request).execute()) {
72              if (!response.isSuccessful()) {
73                  throw new IOException("Error en la respuesta de OpenAI: " + response);
74              }
75
76              try (ResponseBody responseBody = response.body()) {
77                  if (responseBody == null) throw new IOException("Respuesta vacía de OpenAI.");
78
79                  JsonNode jsonResponse = objectMapper.readTree(responseBody.string());
80              }
81          }
82      }
83
84      private String construirPrompt(String pregunta) {
85          // Implementación del método
86      }
87
88      private String generarJsonRequestDesdePrompt(String prompt) {
89          // Implementación del método
90      }
91
92      private void guardarConsultaSPARQL(String pregunta, String resultado) {
93          // Implementación del método
94      }
95
96      private void limpiarCache() {
97          // Implementación del método
98      }
99  }
```

Esta clase es un servicio de Spring (`@Service`) que se encarga de comunicarse con la API de OpenAI. Su función principal es generar consultas SPARQL válidas a partir de preguntas formuladas en lenguaje natural por el usuario.

La clase utiliza inyección de propiedades externas mediante `@Value`, lo que permite parametrizar aspectos como:

- `apiKey`: Clave de acceso a la API de OpenAI.
- `apiUrl`: URL del endpoint de OpenAI.

- *modelo*: Modelo a utilizar (por ejemplo: gpt-3.5-turbo).
- *Temperatura*: Controla la aleatoriedad de las respuestas.
- *maxTokens*: Número máximo de tokens por respuesta.
- *cacheSize*: Tamaño máximo del mapa de caché para preguntas/respuestas ya procesadas.

El método `generarConsultaSPARQL(String pregunta)` es el núcleo del servicio. Utiliza OkHttpClient para hacer la llamada HTTP con Authorization: Bearer y un POST JSON. Realiza los siguientes pasos:

- Consulta la caché local para evitar repetir llamadas si ya se ha procesado esa pregunta.
- Construye un prompt usando la pregunta del usuario.
- Genera un cuerpo JSON con ese prompt y lo envía a OpenAI.
- Procesa la respuesta devolviendo únicamente la consulta SPARQL limpia y validada.
- Almacena la respuesta en caché si es válida.

El método `limpiarConsulta(String consulta)` aplica una limpieza de seguridad y formato sobre la consulta generada:

- Elimina palabras sobrantes como sparql.
- Reemplaza comillas especiales por comillas estándar.
- Añade comillas a tokens mal cerrados o ambiguos.
- Aplica reglas personalizadas para mejorar la sintaxis de la SPARQL antes de ejecutarla.

#### `cacheSPARQL`

Caché LRU (Least Recently Used) implementada como `LinkedHashMap`. Evita llamadas redundantes y mejora el rendimiento. Si se alcanza el tamaño máximo (`cacheSize`), se eliminan las entradas más antiguas.

#### `cargarPromptDatos()`

Lee desde el archivo `contexto-general.txt` (en `resources/prompts/`) el contenido base del prompt.

#### `construirPrompt()`

Añade la pregunta del usuario a ese contexto para construir el mensaje final que se enviará a OpenAI.

En definitiva, *OpenAiService* representa el puente semántico entre el lenguaje natural y el lenguaje formal SPARQL. Su diseño modular, flexible y extensible permite adaptarla a distintos dominios si se cambia el contenido del prompt y la ontología correspondiente.

#### A.4 SparqlService.java (Servicio de ejecución de consultas SPARQL)

```
② SparqlService.java ×
13 import java.net.HttpURLConnection;
14 import java.net.URL;
15 import java.net.URLEncoder;
16
17 import java.util.*;
18
19 @Service 3 usages new *
20 public class SparqlService {
21
22     private static final ObjectMapper objectMapper = new ObjectMapper(); 1 usage
23
24     /**
25      * Ejecuta una consulta SPARQL dinámica contra el endpoint de Ontop.
26      *
27      * @param sparql Consulta SPARQL generada a partir del lenguaje natural.
28      * @return Mapa con los resultados obtenidos desde Oracle, estructurados por variable.
29      * @throws IOException si falla la conexión o el parseo de respuesta.
30     */
31
32     // Ejecución consultas Ontop dinámicas
33     public Map<String, Object> ejecutarConsultaOntop(String sparql) throws IOException { 2 usages new *
34         System.out.println("CONSULTA ENVIADA A ONTOP:\n" + sparql); // LOG
35
36         String endpointUrl = "http://localhost:8080/sparql";
37         URL url = new URL(spec: endpointUrl + "?query=" + URLEncoder.encode(sparql, enc: "UTF-8"));
38         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
39         connection.setRequestMethod("GET");
40         connection.setRequestProperty("Accept", "application/sparql-results+json");
41
42         InputStream inputStream = connection.getInputStream();
43         StringBuilder response = new StringBuilder();
44         try (BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream, charsetName: "UTF-8"))) {
45             String line;
46             while ((line = reader.readLine()) != null) {
47                 response.append(line);
48             }
49         }
    }
```

Este servicio es el encargado de ejecutar consultas SPARQL contra el motor Ontop, que actúa como intermediario entre la ontología y la base de datos Oracle. Ha sido diseñado para ofrecer flexibilidad y permitir la consulta dinámica del sistema mediante preguntas en lenguaje natural.

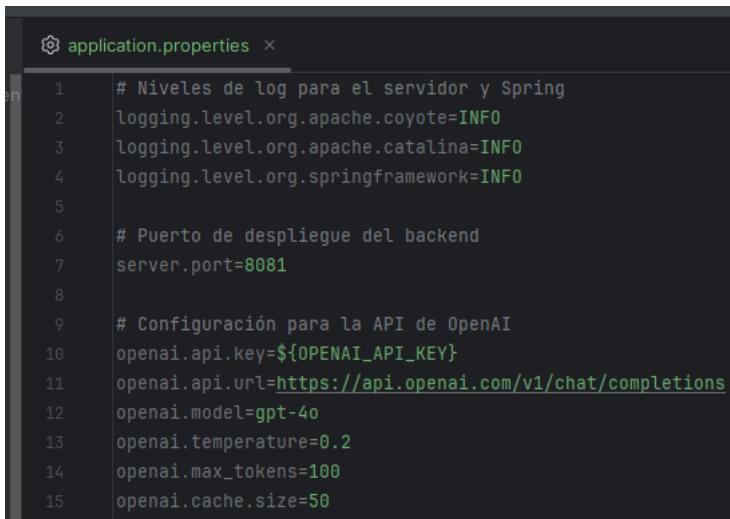
Las dependencias utilizadas son: ObjectMapper de Jackson para parseo de JSON, HttpURLConnection para establecer conexión con el endpoint SPARQL de Ontop, y estructuras estándar de Java (Map, List, etc.).

El método ejecutarConsultaOntop ejecuta una consulta SPARQL generada dinámicamente (por OpenAI) y devuelve los resultados desde Ontop en formato

estructurado. El parámetro *sparql*, consulta en lenguaje SPARQL, ya validada y limpia. La salida un Map con clave "resultado" y una lista de filas, donde cada fila es un Map<String, String> con las variables devueltas por la consulta.

El proceso se traduce en codificar la consulta en URL (GET), a continuación lanza la petición HTTP al endpoint de Ontop (<http://localhost:8080/sparql>). Recoge y parsea la petición en JSON y extrae los valores de cada variable y los devuelve como lista de resultados.

#### A.5 application.properties (Archivo de configuración)



```
# Niveles de log para el servidor y Spring
logging.level.org.apache.coyote=INFO
logging.level.org.apache.catalina=INFO
logging.level.org.springframework=INFO

# Puerto de despliegue del backend
server.port=8081

# Configuración para la API de OpenAI
openai.api.key=${OPENAI_API_KEY}
openai.api.url=https://api.openai.com/v1/chat/completions
openai.model=gpt-4o
openai.temperature=0.2
openai.max_tokens=100
openai.cache.size=50
```

Define la configuración básica del backend Spring Boot, incluyendo los niveles de log, el puerto del servidor, y las variables necesarias para la integración con la API de OpenAI. Este archivo permite una gestión flexible y desacoplada de la configuración, evitando tener que modificar el código fuente directamente.

#### A.6 CorsConfig.java

```

1 package com.proyecto.ayuda.config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.web.servlet.config.annotation.CorsRegistry;
6 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
7
8 @Configuration
9 public class CorsConfig {
10
11     @Bean
12     public WebMvcConfigurer corsConfigurer() {
13         return new WebMvcConfigurer() {
14             @Override
15             public void addCorsMappings(CorsRegistry registry) {
16                 registry.addMapping(pathPattern: "/api/**") // Aplica CORS solo a las rutas de la API
17                     .allowedOrigins("http://localhost:5173") // Permite solicitudes desde React
18                     .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") // Métodos permitidos
19                     .allowedHeaders("*") // Permite todos los headers
20                     .allowCredentials(true); // Permite cookies y autenticación
21             }
22         };
23     }
24 }

```

Es una configuración personalizada de CORS (Cross-Origin Resource Sharing), necesaria para permitir que el frontend desarrollado en React (localhost:5173) pueda comunicarse correctamente con el backend en Spring Boot (localhost:8081).

Sin esta clase, el navegador bloquearía por defecto las peticiones entre orígenes distintos (por política de seguridad del mismo dominio).

## A.7 Archivo pom.xml del proyecto

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>org.avj</groupId>
<artifactId>AyudaInteligente</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
    <java.version>1.8</java.version>
    <spring-boot.version>2.7.14</spring-boot.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
    <!-- Spring Boot (Web, Core, JDBC) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <version>${spring-boot.version}</version>
        <exclusions>
            <exclusion>

```

Define las dependencias, configuración de compilación y plugins del proyecto backend basado en Spring Boot. A continuación se detalla su contenido:

#### Estructura general

- GroupId: org.avj
- ArtifactId: AyudalInteligente
- Versión: 1.0-SNAPSHOT
- Java: Versión 8 (para garantizar compatibilidad con entornos legacy si se requiere).

#### Dependencias incluidas

- Spring Boot 2.7.14  
Compatible con Java 8, se utiliza para lanzar la aplicación (starter, starter-web, starter-jdbc, starter-test).
- Apache POI 5.2.3  
Utilizado para la generación de archivos Excel en el módulo de exportación.  
Compatible con el motor de ayuda inteligente.
- Oracle JDBC (ojdbc8)  
Permite conectar el sistema a la base de datos Oracle mediante el driver oficial.  
Necesaria para que Ontop acceda a los datos relacionales.
- OkHttp 4.11.0  
Cliente HTTP eficiente y moderno que se utiliza para interactuar con la API de OpenAI (modelo gpt-4o), central en la generación dinámica de consultas SPARQL.
- Jackson 2.13.4  
Librería estándar para serialización y deserialización de objetos JSON, necesaria tanto para procesar la respuesta de OpenAI como los resultados SPARQL.
- Spring Boot Test  
Permite incluir pruebas unitarias y de integración.

#### A.8 Archivo *prompt contexto\_general.txt*

```

contexto-general.txt *
1   Eres un generador de consultas SPARQL para un sistema basado en una ontología del dominio "Atención al Ciudadano".
2   Tu tarea es transformar preguntas en lenguaje natural en consultas SPARQL válidas y ejecutables sobre datos reales accedidos mediante Ontop. ✓16
3   La base de datos contiene información sobre:
4     - Personas.
5     - Departamentos.
6     - Registros.
7     - Entidad locales.
8     - Trámites con fase inicio, fase instrucción tiene subfase de alegación, fase fin tiene subfase de justificación y cobro.
9     - Servicios.
10    - Empleo público con etapas.
11    - Criterios de búsqueda.
12    - Operaciones.
13    - Ayuda campos.
14
15 Clases:
16    - :ayudaCampo
17    - :criterioBusqueda
18    - :departamento
19    - :entidadLocal
20    - :empleoPublico
21    - :etapa
22    - :tramiteServicio
23    - :faseInicio
24    - :faseInstruccion
25    - :subfaseAlegacion
26    - :faseFin
27    - :subfaseJustCobro
28    - :servicio
29    - :operacion
30    - :persona
31    - :registro
32
33 Propiedades de ayudaCampo:
34    - :ayuCodigo (xsd:string)
35    - :ayuEmpleado (xsd:string)
36    - :ayuModulo (xsd:string)
37    - :ayuNombre (xsd:string)
38    - :ayuObliga (xsd:string)
39    - :ayuReglas (xsd:string)
40    - :ayuMnemon (xsd:string)

```

## Documentación del Frontend

### A.9 BotonAyuda.jsx

```

  BotonAyuda.jsx ×
1 import React, { useState, useRef, useEffect } from "react";
2 import { motion, AnimatePresence } from "framer-motion";
3 import { MessageCircle, X } from "lucide-react";
4 import Draggable from "react-draggable";
5 import axios from "axios";
6
7 const consultarAyuda = async (pregunta) => {
8   try {
9     const response = await axios.get(
10       `http://localhost:8081/api/ayuda/consulta?pregunta=${encodeURIComponent(pregunta)}`
11     );
12     const datos = response.data.resultado;
13
14     console.log("👉 Datos recibidos:", datos);
15
16     if (Array.isArray(datos) && datos.length > 0) {
17       return datos;
18     } else {
19       return "No se encontró información relevante.";
20     }
21   } catch (error) {
22     console.error("ERROR:", error.response ? error.response.data : error.message);
23     return "Error al consultar la información.";
24   }
25 };
26
27 const BotonAyuda = () => {
28   const [mensaje, setMensaje] = useState("");
29   const [mensajes, setMensajes] = useState(() => {
30     const mensajesGuardados = localStorage.getItem("chatMensajes");
31     return mensajesGuardados ? JSON.parse(mensajesGuardados) : [];
32   });

```

Ha sido desarrollado en React con la librería framer-motion para animaciones, axios para realizar llamadas HTTP y react-draggable para permitir el movimiento libre del modal en la pantalla. Su objetivo es proporcionar al usuario una interfaz interactiva desde la cual consultar dudas o generar listados personalizados a partir de lenguaje natural.

*mensaje*: almacena el texto introducido por el usuario.

*mensajes*: guarda el historial de conversación, persistido localmente en localStorage.

*mostrarChat*: controla la visibilidad del modal.

*mostrarBotonDescarga*: se activa si la respuesta es un conjunto tabular exportable.

*ultimaPregunta*: necesaria para volver a usar la pregunta en la exportación a Excel.

*textAreaRef* y *dragRef*: referencias para ajustar dinámicamente el área de texto y permitir el movimiento del modal.

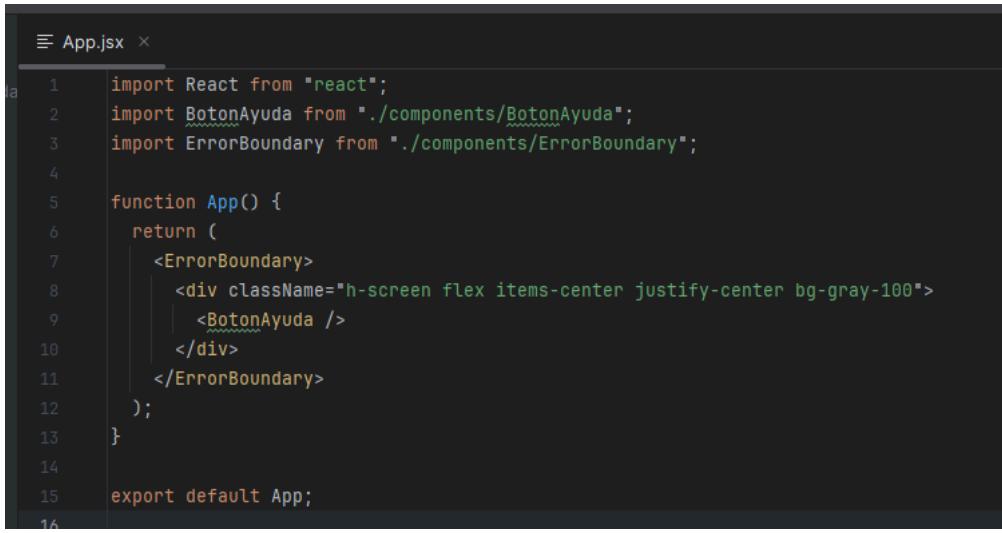
## A.10 ErrorBoundary.jsx

```
≡ ErrorBoundary.jsx ×  
1 import React, { Component } from "react";  
2  
3 class ErrorBoundary extends Component {  
4   constructor(props) {  
5     super(props);  
6     this.state = { hasError: false };  
7   }  
8  
9   static getDerivedStateFromError(error) {  
10     return { hasError: true };  
11   }  
12  
13   componentDidCatch(error, errorInfo) {  
14     console.error("Error en el componente:", error, errorInfo);  
15   }  
16  
17   render() {  
18     if (this.state.hasError) {  
19       return (  
20         <div className="p-4 bg-red-100 text-red-700 rounded">  
21           <h2>Algo salió mal!</h2>  
22           <p>Por favor, intenta recargar la página.</p>  
23         </div>  
24       );  
25     }  
26     return this.props.children;  
27   }  
28 }  
29  
30 export default ErrorBoundary;  
31 }
```

Para asegurar una experiencia robusta en la interfaz de usuario, se ha implementado un componente *ErrorBoundary*, una técnica estándar en React para capturar errores en tiempo de ejecución en los componentes hijos.

Este componente actúa como una “capa de seguridad” alrededor del botón de ayuda u otros módulos, mostrando un mensaje amable en caso de que se produzca un fallo inesperado, en lugar de romper la aplicación completa. Su uso mejora la resiliencia y el control del flujo de errores desde el punto de vista visual.

## A.11 App.jsx



```
App.jsx ×
1 import React from "react";
2 import BotonAyuda from "./components/BotonAyuda";
3 import ErrorBoundary from "./components/ErrorBoundary";
4
5 function App() {
6   return (
7     <ErrorBoundary>
8       <div className="h-screen flex items-center justify-center bg-gray-100">
9         <BotonAyuda />
10      </div>
11    </ErrorBoundary>
12  );
13}
14
15 export default App;
16
```

Este archivo es el punto de entrada principal del frontend en React. Su función es renderizar la aplicación en pantalla.

El código completo del backend y frontend, así como el prompt, los archivos de configuración y el archivo .properties, se encuentran disponibles en el repositorio de GitHub del proyecto:

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/backend>

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/frontend>

## Anexo IV. Archivo .obda y Ontop

Este anexo documenta el archivo de mapeo OBDA (.obda) utilizado para vincular los conceptos definidos en la ontología OWL del sistema con las estructuras

reales de la base de datos Oracle mediante Ontop. Así como, la instalación de Ontop en el Docker.

### Documentación archivo .obda

El archivo .obda (Ontology-Based Data Access) especifica cómo se traducen las consultas SPARQL realizadas sobre la ontología a instrucciones SQL ejecutables sobre una base de datos relacional. En este caso, la tecnología utilizada ha sido Ontop, una herramienta que permite realizar este mapeo de manera declarativa y eficiente.

El mapeo incluye tres componentes clave:

- PrefixDeclaration: Define los prefijos utilizados (ej. xsd, :).
- MappingDeclaration: Contiene las reglas de transformación de SPARQL a SQL.
- target/source: Especifican la estructura RDF generada (target) y la consulta SQL (source) desd

El archivo contiene múltiples bloques mappingId, cada uno de los cuales:

- Representa una correspondencia entre una tabla o vista de Oracle y una clase o propiedad de la ontología.
- Permite que cada instancia en la base de datos quede representada como una instancia RDF accesible vía SPARQL.

Por ejemplo:

```
mappingId    personaNombre
target       :persona_{PER_CODIGO} :perNombre "{PER_NOMBRE}" .
source        SELECT PER_CODIGO, PER_NOMBRE FROM SYSTEM.AC_PERSONAS
```

Este bloque permite que una persona registrada en Oracle se represente con su nombre semánticamente a través de la propiedad :perNombre en la ontología.

El archivo completo .obda incluye mapeos para las siguientes entidades:

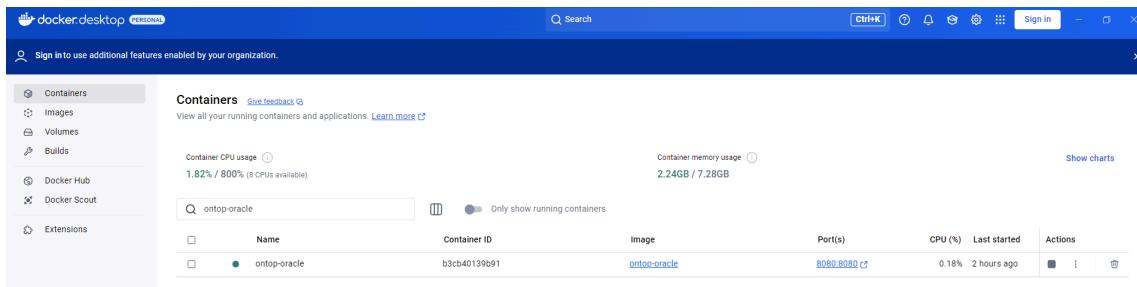
- Personas
- Departamentos
- Registros
- Entidades Locales
- Empleo Público

- Trámites y Servicios.
- Fases y subfases de trámites
- Etapas y fases de empleo público
- Servicios
- Operaciones
- Campos de Ayuda y Criterios de Búsqueda

Cada uno de estos elementos está vinculado semánticamente para garantizar que las consultas SPARQL devuelvan datos consistentes, estructurados y semánticamente ricos.

### Instalación de Ontop con Docker

Ontop se instala y ejecuta dentro de un contenedor Docker, asegurando portabilidad y facilidad de configuración. El sistema se conecta al contenedor de Oracle dentro de la misma red de Docker.



El archivo completo de mapeo .obda, así como los recursos relacionados con la ontología, están disponibles en el repositorio de GitHub del proyecto:

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/odba>

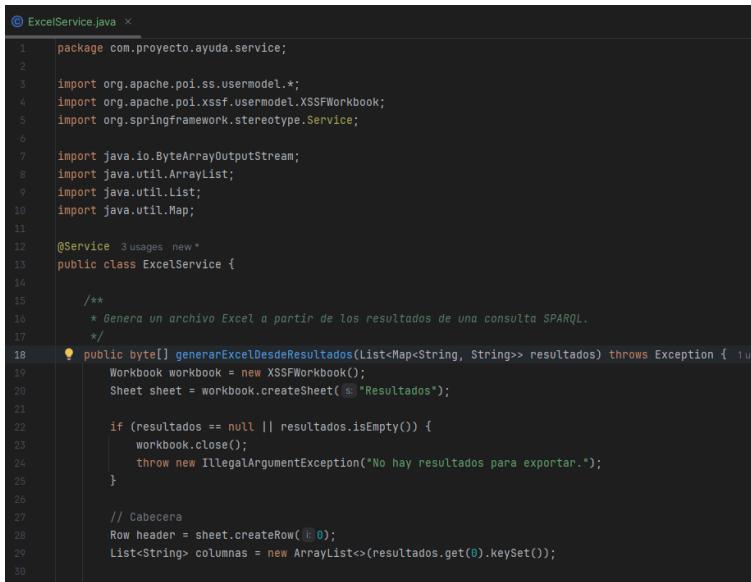
## Anexo V. Doc. Backend y frontend “Consultas personalizadas”

Este anexo describe la implementación del servicio de exportación de resultados en formato Excel (.xlsx) dentro del sistema de ayuda inteligente. Esta

funcionalidad permite al usuario descargar listados estructurados generados por consultas SPARQL, facilitando su análisis externo o almacenamiento.

## Documentación del Backend (Spring Boot)

### A.12 ExcelService.java



```
1 package com.proyecto.ayuda.service;
2
3 import org.apache.poi.ss.usermodel.*;
4 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
5 import org.springframework.stereotype.Service;
6
7 import java.io.ByteArrayOutputStream;
8 import java.util.ArrayList;
9 import java.util.List;
10 import java.util.Map;
11
12 @Service 3 usages new*
13 public class ExcelService {
14
15     /**
16      * Genera un archivo Excel a partir de los resultados de una consulta SPARQL.
17      */
18     public byte[] generarExcelDesdeResultados(List<Map<String, String>> resultados) throws Exception { 1us
19         Workbook workbook = new XSSFWorkbook();
20         Sheet sheet = workbook.createSheet( 2 "Resultados");
21
22         if (resultados == null || resultados.isEmpty()) {
23             workbook.close();
24             throw new IllegalArgumentException("No hay resultados para exportar.");
25         }
26
27         // Cabecera
28         Row header = sheet.createRow( 3 0);
29         List<String> columnas = new ArrayList<>(resultados.get(0).keySet());
30     }
}
```

Tiene como objetivo de facilitar al usuario la descarga de resultados generados por el sistema en formato estructurado, se ha implementado un nuevo componente dentro del backend. Este módulo permite exportar automáticamente la información obtenida mediante consultas SPARQL a un archivo Excel (.xlsx).

El servicio *ExcelService* cumple las siguientes funciones:

- Recibe desde el controlador (*AyudaController.java*) los datos resultantes de la consulta SPARQL.
- Identifica si el resultado es un listado estructurado que puede representarse en formato tabular.
- Genera dinámicamente un archivo Excel utilizando la librería *Apache POI*.
- Devuelve el archivo al frontend como una respuesta binaria (*application/vnd.ms-excel*) lista para descarga.

Este servicio se activa únicamente cuando la respuesta contiene una estructura de tipo array de objetos JSON, lo que garantiza que se exporta únicamente información relevante y organizada.

La exportación se realiza a través del endpoint */api/ayuda/exportar*, accesible desde el frontend.

El código principal del servicio está dividido en los siguientes métodos:

- `generarExcelDesdeResultados()`: construye el documento Excel a partir de una lista de objetos JSON.
- `exportarExcel()`: expone el endpoint para descarga desde el frontend.
- Métodos auxiliares para validación, formateo de celdas y creación de cabeceras.

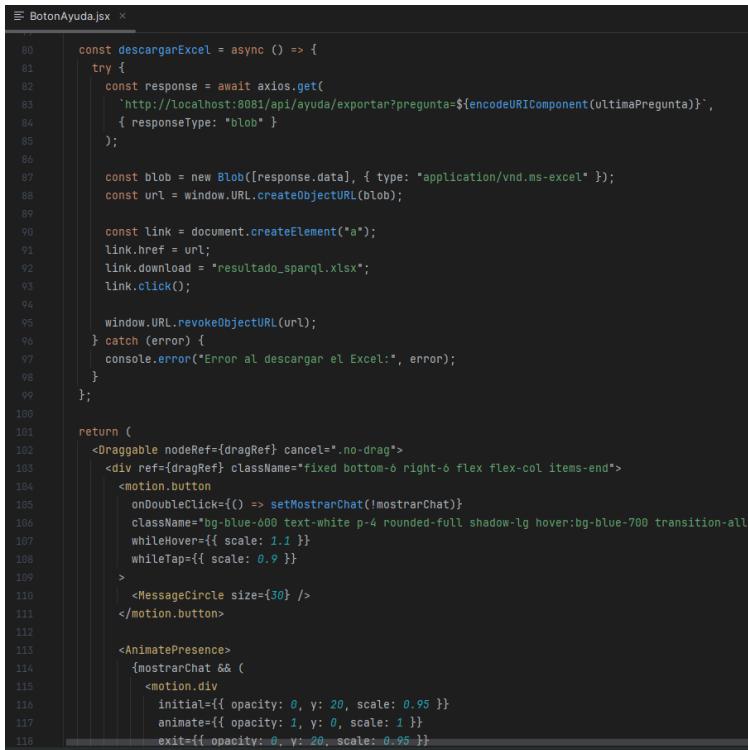
Este diseño modular permite extender fácilmente la lógica para aplicar estilos, formatos condicionales o incluir metadatos en los informes exportados.

Todo el código relacionado con este servicio se encuentra disponible en el repositorio de GitHub del proyecto:

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/backend>

## Documentación del Frontend

### A.13 BotonAyuda.jsx



```
const descargarExcel = async () => {
  try {
    const response = await axios.get(
      `http://localhost:8081/api/ayuda/exportar?pregunta=${encodeURIComponent(UltimaPregunta)}`,
      { responseType: "blob" }
    );

    const blob = new Blob([response.data], { type: "application/vnd.ms-excel" });
    const url = window.URL.createObjectURL(blob);

    const link = document.createElement("a");
    link.href = url;
    link.download = "resultado_sparql.xlsx";
    link.click();

    window.URL.revokeObjectURL(url);
  } catch (error) {
    console.error("Error al descargar el Excel:", error);
  }
};

return (
  <Draggable nodeRef={dragRef} cancel=".no-drag">
    <div ref={dragRef} className="fixed bottom-0 right-0 flex flex-col items-end">
      <motion.button
        onDoubleClick={() => setMostrarChat(!mostrarChat)}
        className="bg-blue-600 text-white p-4 rounded-full shadow-lg hover:bg-blue-700 transition-all"
        whileHover={{ scale: 1.1 }}
        whileTap={{ scale: 0.9 }}
      >
        <MessageCircle size={30} />
      </motion.button>

      <AnimatePresence>
        {mostrarChat && (
          <motion.div
            initial={{ opacity: 0, y: 20, scale: 0.95 }}
            animate={{ opacity: 1, y: 0, scale: 1 }}
            exit={{ opacity: 0, y: 20, scale: 0.95 }}
          >
        )}
      </AnimatePresence>
    </div>
  </Draggable>
);
```

Se añade la lógica de comunicación con el backend (API REST desarrollada en Sprint Boot) la descarga de resultados estructurados como Excel (/api/ayuda/exportar).

Proceso de exportación:

- El usuario formula una pregunta que genera una respuesta estructurada (por ejemplo, un listado de personas o trámites).
- El sistema detecta que la respuesta es una tabla (array de objetos).
- Se activa un botón adicional: “Descargar Excel”.
- Al hacer clic, se realiza una llamada GET al endpoint /api/ayuda/exportar, pasando como parámetro la misma pregunta original.
- El backend procesa esta petición y genera un archivo .xlsx utilizando Apache POI.
- El frontend descarga automáticamente el archivo al navegador del usuario.

El código fuente del componente se encuentra en el repositorio de GitHub del proyecto:

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/frontend>

## Anexo VI. Evaluación “Botón de Ayuda Inteligente”

Este anexo recoge los resultados completos obtenidos durante el proceso de validación del botón de ayuda inteligente implementado en el sistema. El objetivo de esta evaluación ha sido comprobar si el sistema es capaz de proporcionar respuestas útiles, precisas y contextualizadas ante consultas realizadas por el usuario, apoyándose en una ontología del dominio y en datos ficticios extraídos de la base de datos mediante consultas SPARQL.

La evaluación se ha estructurado en iteraciones, cada una de ellas compuesta por un conjunto de preguntas clasificadas en función de su nivel de dificultad: básica, intermedia o avanzada. Para cada pregunta se ha documentado la respuesta obtenida, su grado de relevancia y exactitud, el tiempo de respuesta, la necesidad de reformulación, la sección funcional evaluada y otras observaciones relevantes.

A continuación, se presentan las tablas con los resultados correspondientes a cada iteración realizada.

**Primera y segunda iteración cálculo de tiempo respuesta (s):**

Pregunta	Respuesta	¿Relevante?	Exactitud	Tiempo Respuesta (s)	¿Reformulación?	Nivel Dificultad	Sección Evaluada	Observaciones
¿Cómo tengo que escribir el nombre completo de la persona?	Solo letras. No debe estar vacío. Ejemplo: María Dolores	Sí	Alta	0.02 0.01	No	Básica	Personas	Respuesta directa y sin ambigüedad.
¿Qué formato debe tener el número de identificación?	8 números seguidos de una letra. Ejemplo: 12345678A.	Sí	Alta	5.79 0.02	No	Básica	Personas	Muy clara. Incluye formato y ejemplo.
¿Es obligatorio poner los dos apellidos?	El segundo apellido es opcional. Solo el primero es obligatorio.	Sí	Alta	1.15 0.02	No	Básica	Personas	El sistema entiende bien la naturaleza legal.
¿Qué tengo que poner en la función de la persona?	Texto libre. Ejemplo: Técnico de sistemas	Sí	Alta	1.39 0.02	No	Medio	Personas	La respuesta combina definición, ejemplo y reglas.
¿Cuál es la diferencia entre el teléfono interno y el externo?	No se encontró información relevante.	No	-	0.92 0.92	No	Medio	Personas	No está registrado en la tabla de ayuda.
¿Qué significa "adscrito a un puesto"? ¿Tengo que marcarlo?	Indica si la persona está adscrita a un puesto. Ejemplo: Sí	Sí	Alta	1.65 0.01	No	Avanzado	Personas	Buena comprensión semántica de un campo booleano.
¿A qué se refiere "unidad administrativa"? ¿Cómo la selecciono?	Unidad administrativa a la que pertenece la persona. Ejemplo: Unidad TIC	Sí	Alta	2.70 0.01	No	Avanzado	Personas	Excelente respuesta con definición, ejemplo y validación.

**Primera y segunda iteración cálculo de tiempo respuesta (s):**

Pregunta	Respuesta	¿Relevante?	Exactitud	Tiempo Respuesta (s)	¿Reformulación?	Nivel Dificultad	Sección Evaluada	Observaciones
¿Cómo debo indicar el horario de atención si varía por días?	Texto libre. Máximo 100 caracteres. Horario de atención al ciudadano del departamento. Ejemplo: lunes a viernes, 9:00-14:00	Sí	Alta	1.90 0.02	No	Avanzado	Registros	Respuesta clara, con ejemplo práctico. El sistema identifica correctamente la naturaleza del campo.
¿Qué pasa si no conozco el departamento al que pertenece el registro?	No se encontró información relevante.	No	-	0.92 0.92	No	Avanzado	Registros	Falta información en la tabla de ayuda para este campo; convendría ampliarla con orientaciones para el usuario.
¿Cómo se especifica una entidad local que no pertenece a una provincia?	No se encontró información relevante.	No	-	0.92 0.92	No	Avanzado	Entidad local	No está contemplado el caso en la tabla de ayuda; sería útil incluirlo para cubrir situaciones excepcionales.
¿Cómo introduzco un departamento que ya no existe pero necesito registrar por motivos históricos?	Texto libre. Ejemplo: Técnico de sistemas	Sí	Alta	1.70 0.02	No	Avanzado	Departamentos	El sistema interpreta correctamente la pregunta y ofrece una respuesta flexible, aunque algo genérica.
¿Para qué se usa el fax externo en un registro?	Formato válido. Máximo 20 caracteres. Número de fax externo del registro. Para comunicaciones desde y hacia el exterior de la administración. Ejemplo: 963112235	Sí	Alta	1.72 0.02	No	Medio	Registros	La respuesta incluye uso, longitud y ejemplo. Muy útil para usuarios sin conocimientos técnicos.
¿Qué datos tengo que poner en el campo "comarca" de una entidad local?	Texto libre. Opcional. Comarca a la que pertenece la entidad local. Ejemplo: Horta Sud	Sí	Alta	2.49 0.02	No	Medio	Entidad local	La ayuda devuelve información muy completa, incluyendo obligatoriedad y ejemplo claro.
¿Qué significa el campo "tipo" dentro de	Tipo de centro (ej. Oficina, Registro, etc.). Ejemplo: Oficina	Sí	Alta	1.20 0.02	No	Medio	Departamentos	Respuesta directa y útil. El campo es entendible para

los datos de un departamento?								usuarios sin formación previa.
-------------------------------	--	--	--	--	--	--	--	--------------------------------

**Primera y segunda iteración cálculo de tiempo respuesta (s):**

Pregunta	Respuesta	¿Relevante?	Exactitud	Tiempo Respuesta (s)	¿Reformulación?	Nivel Dificultad	Sección Evaluada	Observaciones
¿Cómo debo escribir la descripción del trámite para que sea clara para los ciudadanos?	Texto libre. Máximo 500 caracteres. Descripción detallada del trámite o servicio. Ejemplo: Solicitud de becas para estudios universitarios	Sí	Alta	1.88 0.02	No	Medio	Trámite y servicios	La respuesta incluye el tipo de contenido esperado, longitud y un ejemplo claro. Muy útil para el usuario.
¿Qué pasa si la fecha de fin del plazo para presentar la solicitud cae en festivo?	No se encontró información relevante.	No	-	0.92 0.92	No	Avanzado	Fase de inicio	No devuelve información. Este tipo de validación no está contemplada en la ayuda semántica. Se puede registrar como futura mejora.
¿Dónde se puede consultar la normativa aplicable durante la fase de instrucción?	<a href="http://example.org/instrucion-tramite2">http://example.org/instrucion-tramite2</a>	Sí	Alta	0.95 0.02	Sí	Medio	Fase de instrucción	Se devuelve correctamente el enlace registrado en la ayuda. El sistema reconoce bien el campo de normativa. Se reformularía para que devuelva solo 1.
¿Qué formato debe tener la fecha de inicio de la justificación y cobro?	Formato de fecha YYYY-MM-DD. Fecha de inicio de la subfase de justificación y cobro. Ejemplo: 2025-07-01	Sí	Alta	1.80 0.01	No	Medio	Subfase de justificación y cobro	Responde con el formato esperado y da ejemplo claro. Se entiende bien incluso sin nombrar el campo técnico.
En la fase fin ¿Qué información debo incluir en el campo de procedimiento de cobro si el trámite tiene tasas?	Texto libre. Máximo 500 caracteres. Procedimiento de cobro asociado a la fase de finalización. Ejemplo: Transferencia bancaria	Sí	Alta	2.68 0.03	Sí	Avanzado	Fase fin	Requiere reformulación incluyendo explícitamente "fase fin" para obtener la respuesta esperada. Mejora posible en el reconocimiento contextual.

¿Es obligatorio llenar la forma de presentación de una alegación o puede dejarse en blanco?	No	Sí	Alta	1.82 0.02	No	Avanzado	Subfase de alegación	El sistema responde correctamente sobre obligatoriedad. Muy útil en escenarios de validación de datosopcionales.
---	----	----	------	--------------	----	----------	----------------------	--

### Primera y segunda iteración cálculo de tiempo respuesta (s):

Pregunta	Respuesta	¿Relevante?	Exactitud	Tiempo Respuesta (s)	¿Reformulación?	Nivel Dificultad	Sección Evaluada	Observaciones
¿Qué criterios de valoración deben especificarse en la fase de instrucción de un trámite?	Baremo de criterios objetivos publicados previamente. Evaluación técnica y adecuación a la normativa. Baremo de criterios objetivos publicados previamente. Puntuación basada en experiencia y méritos. Baremo de criterios objetivos publicados previamente.	Sí	Alta	1.52 0.02	Sí	Avanzado	Fase instrucción	Muestra todos los criterios. Se reformula la pregunta indicándole un límite de 6.
¿Cómo puedo detallar el departamento que tramita el procedimiento en la fase de instrucción?	Texto libre. Máximo 200 caracteres. Departamento responsable de la tramitación del trámite. Ejemplo: Departamento de Evaluación	Sí	Alta	1.35 0.02	Sí	Avanzado	Fase instrucción	Se reformula indicando que es en la fase instrucción. Se observa que se confunde con la palabra trámites y servicios al hacer la consulta SPARQL. Se puede entrenar reforzando el prompt.
¿Qué debo incluir en el campo de requisitos de un servicio?	Texto libre. Máximo 500 caracteres. Requisitos necesarios para utilizar el servicio. Ejemplo: Estar empadronado en la comunidad	Sí	Alta	1.61 0.02	No	Medio	Servicios	Información coherente y exacta.
¿Cómo se define el objeto de un servicio?	Texto libre. Máximo 500 caracteres. Objeto del servicio que describe su finalidad.	Sí	Alta	1.75 0.02	No	Medio	Servicios	Información coherente y exacta.

¿Qué tipo de información complementaria puedo añadir en un servicio?	Texto libre. Máximo 500 caracteres. Información complementaria del servicio. Ejemplo: Se recomienda solicitar cita previa	Sí	Alta	2.21 0.02	No	Avanzada	Servicios	Información coherente y exacta.
¿Qué formato debe tener la fecha de inicio del servicio?	Formato de fecha YYYY-MM-DD. Fecha de inicio del servicio. Ejemplo: 2025-04-01	Sí	Alta	1.72 0.02	No	Avanzada	Servicios	Información coherente y exacta.
¿Cómo se indica el tipo de personal al que va dirigida una convocatoria?	Debe ser Funcionario, Laboral, u otro tipo reconocido. Tipo de personal al que se dirige la oferta (funcionario, laboral, etc). Ejemplo: Funcionario	Sí	Alta	2.49 0.02	No	Medio	Empleo público	Información coherente y exacta.
¿Qué tengo que poner si la etapa aún no ha comenzado pero ya está definida?	No se encontró información relevante.	No	-	0.92 0.92	No	Avanzado	Etapa empleo público	No devuelve información. Este tipo de validación no está contemplada en la ayuda semántica. Se puede registrar como futura mejora.
¿Qué tipo de comodines puedo usar al buscar por nombre?	VALOR%	Sí	Alta	0.98 0.02	No	Medio	Criterio búsqueda	Información coherente y exacta.
¿Cómo sé si un criterio de búsqueda está disponible en varias secciones?	No se encontró información relevante.	No	-	0.92 0.92	No	Avanzado	Criterio búsqueda	No devuelve información. Este tipo de validación no está contemplada en la ayuda semántica. Se puede registrar como futura mejora.
¿Para qué sirve la operación “Insertar” en un trámite o servicio?	No se encontró información relevante.		-	1.15 0.92	No	Medio	Operación	No devuelve información ya que considera las comillas dobles. Se ha de indicar en el diccionario establecido en el prompt y volver a probar.
¿Es posible aplicar una operación de modificación a una entidad que no ha sido creada previamente?	No se encontró información relevante.		-	1.13 0.76	No	Avanzado	Operación	Tipo de preguntas avanzadas que aplican lógica no las resuelve. Tarea futura.



## Anexo VII. Evaluación “Detección de errores”

Este anexo recoge el detalle completo de la evaluación realizada sobre la funcionalidad de detección automática de errores implementada en el sistema.

Inicialmente se han preparado registros manualmente en Oracle para cada uno de los tipos de errores descritos en el apartado 4.1. Quedan reportados en el archivo *SimulaErrores.sql* ubicado en el repositorio compartido GitHub:

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/p4DetectErrores>

Posteriormente se ha realizado el mapeo de los errores simulados en el archivo .odba, reportado en GitHub:

<https://github.com/amvajua/TFM-Ayuda-Inteligente/tree/main/odba>

Imagen AVII.1 Fragmento de *ErrorConsistencia* en el fichero de mapeo .odba

```
mappingId errorTramiteTipo
target :error_tramite_{TRS_CODIGO} a :ErrorConsistencia .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL

mappingId errorTramiteDescripcion
target :error_tramite_{TRS_CODIGO} :errorDescripcion "El trámite no tiene departamento asignado" .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL

mappingId errorTramiteFecha
target :error_tramite_{TRS_CODIGO} :errorFecha "2025-05-01T00:00:00"^^xsd:dateTime .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL

mappingId errorTramiteDetectadoEn
target :error_tramite_{TRS_CODIGO} :errorDetectadoEn :tramite_{TRS_CODIGO} .
source SELECT TRS_CODIGO FROM SYSTEM.AC_TRAMITES_SERVICIOS WHERE TRS_DEPCOD IS NULL
```

Las consultas SPARQL que extraen los diferentes tipos de errores ejecutadas en Ontop se muestran a continuación:

1. Detecta trámites que no están vinculados a ninguna unidad, lo cual representa un error de asignación según la lógica del dominio.

SPARQL: Trámites si departamento asignado

The screenshot shows the Ontop SPARQL endpoint interface. At the top, there is a navigation bar with tabs for Table, Response, Pivot Table, Google Chart, Geo, and a search bar. Below the navigation bar, there is a query editor window containing the following SPARQL code:

```
1: PREFIX : <http://example.org/ontologika>
2:
3: SELECT ?error ?descripcion ?fecha ?afectaA
4: WHERE {
5:   ?error a :ErrorConsistencia ;
6:   ?error :errorDescripcion ?descripcion ;
7:   ?error :errorFecha ?fecha ;
8:   ?error :errorDetectadoEn ?afectaA .
9: }
```

Below the query editor, a table displays the results of the query. The table has four columns: error, descripción, fecha, and afectaA. There is one row of data:

error	descripción	fecha	afectaA
<a href="http://example.org/ontologika/error_tramite_101">http://example.org/ontologika/error_tramite_101</a>	El trámite no tiene departamento asignado	'2025-05-01T00:00:00'^^xsd:dateTime	<a href="http://example.org/ontologika/tramite_101">http://example.org/ontologika/tramite_101</a>

At the bottom of the interface, there is a footer with the text "Showing 1 to 1 of 1 entries (in 0.190 seconds)".

## Ontop SPARQL endpoint

```

Query 18 X   Query 16 X   Query 17 X   Query 15 X
Query 32 X   Query 27 X   Query 25 X   Query 24 X

1 v PREFIX : <http://example.org/ontologia#>
2
3 SELECT ?error ?descripcion ?fecha ?afectaA
4 WHERE {
5   ?error a :ErrorConsistencia ;
6     :errorDescripcion ?descripcion ;
7     :errorFechaDetecc ?fecha ;
8     :errorDetectadoEn ?afectaA .
9 }
10

```

Showing 1 to 1 of 1 entries (in 0.126 seconds)

error	descripcion
<a href="http://example.org/ontologia#error_tramite_101">http://example.org/ontologia#error_tramite_101</a>	El trámite no tiene departamento asignado

Showing 1 to 1 of 1 entries (in 0.126 seconds)

fecha	afectaA
"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#tramite_101">http://example.org/ontologia#tramite_101</a>

- Permite detectar errores en la información de las personas del sistema, que carecen de vinculación organizativa.

SPARQL: Personas sin departamento asignado

Ontop SPARQL endpoint

Showing 1 to 3 of 3 entries (in 0.051 seconds)

codigoPersona	descripcion	fecha	uriPersona
34	La persona no tiene departamento asignado	"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#persona_34">http://example.org/ontologia#persona_34</a>
201	La persona no tiene departamento asignado	"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#persona_201">http://example.org/ontologia#persona_201</a>
202	La persona no tiene departamento asignado	"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#persona_202">http://example.org/ontologia#persona_202</a>

## Ontop SPARQL endpoint

Query 18 X    Query 16 X    Query 17 X    Query 15 X    Query 14 X  
 Query 34 X    **Query 32 X**    Query 27 X    Query 25 X    Query 24 X

```

1+ PREFIX : <http://example.org/ontologia#>
2+ PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3+
4+ SELECT ?codigoPersona ?descripcion ?fecha ?uriPersona
5+ WHERE {
6+   ?error a :ErrorConsistencia ;
7+   :errorDescripcion ?descripcion ;
8+   :errorFechaDetecc ?fecha ;
9+   :errorDetectadoEn ?uriPersona .
10+
11+ FILTER(STRSTARTS(STR(?error), STR(:error_persona_)))
12+
13+ BIND(STRAFTER(STR(?uriPersona), STR(:persona_)) AS ?codigoPersona)
14+
15+ ORDER BY DESC(?fecha)
16+
17+
    
```

**Table** Response Pivot Table Google Chart Geo </>

Showing 1 to 3 of 3 entries (in 0.051 seconds)

	codigoPersona	descripcion
1	34	La persona no tiene departamento asignado
2	201	La persona no tiene departamento asignado
3	202	La persona no tiene departamento asignado

Showing 1 to 3 of 3 entries (in 0.051 seconds)

Search

fecha	uriPersona
"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#persona_34">http://example.org/ontologia#persona_34</a>
"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#persona_201">http://example.org/ontologia#persona_201</a>
"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#persona_202">http://example.org/ontologia#persona_202</a>

### 3. Identifica errores de consistencia temporal en los datos de los procedimientos.

#### SPARQL: Fechas de finalización anteriores a la fecha de inicio

Ontop SPARQL endpoint

endpoint address: <http://localhost:8080/board> | ontop v5.3.0

Query 18 X    Query 16 X    Query 17 X    Query 15 X    Query 4 X    Query 13 X    Query 12 X    Query 11 X    Query 10 X    Query 9 X    Query 8 X    Query 6 X    Query 7 X    Query X    Query 1 X    Query 3 X    Query 5 X    Query 31 X    Query 26 X  
 Query 35 X    Query 34 X    Query 32 X    Query 27 X    **Query 25 X**    Query 24 X    Query 23 X    Query 22 X    Query 19 X    Query 20 X    Query 21 X    Query 28 X    Query 29 X    Query 30 X    Query 31 X    +

```

1+ PREFIX : <http://example.org/ontologia#>
2+ PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3+
4+ SELECT ?error ?descripcion ?fecha ?faseInicio
5+ WHERE {
6+   ?error a :ErrorConsistencia ;
7+   :errorDescripcion ?descripcion ;
8+   :errorFechaDetecc ?fecha ;
9+   :errorDetectadoEn ?faseInicio .
10+
11+ FILTER(STRSTARTS(STR(?error), STR(:error_faseFinalizaAnteriorAInicio)))
12+
13+ ORDER BY ?faseInicio
14+
    
```

**Table** Response Pivot Table Google Chart Geo </>

Showing 1 to 1 of 1 entries (in 0.214 seconds)

error	descripcion	fecha	faseInicio
<a href="http://example.org/ontologia#Error_faseFinalizaFechas_20">http://example.org/ontologia#Error_faseFinalizaFechas_20</a>	La fecha de finalización es anterior a la fecha de inicio	"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#faseInicio_20">http://example.org/ontologia#faseInicio_20</a>

having 1 to 1 of 1 entries (in 0.214 seconds)

### Ontop SPARQL endpoint

```

Query 18 X   Query 16 X   Query 17 X   Query 15 X   Query 14 X
Query 35 X   Query 34 X   Query 32 X   Query 27 X   Query 25 X

1 PREFIX : <http://example.org/ontologia#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?error ?descripcion ?fecha ?faseInicio
5 WHERE {
6   ?error a :ErrorConsistencia ;
7     :errorDescripcion ?descripcion ;
8     :errorFechaDetec ?fecha ;
9     :errorDetectadoEn ?faseInicio .
10    FILTER(STRSTARTS(STR(?error), STR(:error_faseInicioFechas_)))
11  }
12 ORDER BY ?faseInicio
13

```

**Table** Response Pivot Table Google Chart Geo </>

Showing 1 to 1 of 1 entries (in 0.214 seconds)

	error	descripcion
1	<a href="http://example.org/ontologia#error_faseInicioFechas_20">http://example.org/ontologia#error_faseInicioFechas_20</a>	La fecha de finalización es anterior a la fecha de inicio

Showing 1 to 1 of 1 entries (in 0.214 seconds)

Search:

fecha

"2025-05-01T00:00:00"^^xsd:dateTime

faseInicio

[http://example.org/ontologia#faseInicio\\_20](http://example.org/ontologia#faseInicio_20)

#### 4. Detecta registros incompletos por ausencia de un campo obligatorio.

SPARQL: Campo descripción obligatorio en Empleo público.

Ontop SPARQL endpoint

Query 18 X Query 16 X Query 17 X Query 15 X Query 14 X Query 4 X Query 13 X Query 12 X Query 11 X Query 10 X Query 9 X Query 8 X Query 6 X Query 7 X Query X Query 1 X Query 3 X Query 2 X Query 5 X Query 33 X Query 26 X
Query 35 X Query 34 X Query 32 X Query 27 X Query 25 X Query 24 X Query 23 X Query 22 X Query 19 X Query 20 X Query 21 X Query 28 X Query 29 X Query 30 X Query 31 X +

```

1 PREFIX : <http://example.org/ontologia#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?codigo ?descripcion ?fecha ?detectadoEn
5 WHERE {
6   ?error a :ErrorConsistencia ;
7     :errorDescripcion ?descripcion ;
8     :errorFechaDetec ?fecha ;
9     :errorDetectadoEn ?detectadoEn .
10    FILTER(CONTAINS(STR(?error), "error_empleoDescripcion_"))
11  }
12  BIND(STRAFTER(STR(?error), "error_empleoDescripcion_") AS ?codigo)
13

```

**Table** Response Pivot Table Google Chart Geo </>

Showing 1 to 1 of 1 entries (in 0.62 seconds)

	codigo	descripcion	fecha	detectadoEn
1	51	El campo ERU_DESCRIPCION es obligatorio y está vacío	"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#empleoPublico_51">http://example.org/ontologia#empleoPublico_51</a>

Showing 1 to 1 of 1 entries (in 0.62 seconds)

### Ontop SPARQL endpoint

Query 18 X Query 16 X Query 17 X Query 15 X Query 14 X

Query 36 X Query 35 X Query 34 X Query 32 X Query 27 X

```

1 PREFIX : <http://example.org/ontologia#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?codigo ?descripcion ?fecha ?detectadoEn
5 WHERE {
6   ?error a :ErrorConsistencia ;
7     :errorDescripcion ?descripcion ;
8     :errorFechaDetec ?fecha ;
9     :errorDetectadoEn ?detectadoEn .
10    FILTER(CONTAINS(STR(?error), "error_empleoDescripcion_"))
11  }
12  BIND(STRAFTER(STR(?error), "error_empleoDescripcion_") AS ?codigo)
13

```

Showing 1 to 1 of 1 entries (in 0.62 seconds)		
codigo	descripcion	fecha
1 51	El campo EPU_DESCRIPCION es obligatorio y está vacío	"2025-05-01T00:00:00"^^xsd:dateTime

Showing 1 to 1 of 1 entries (in 0.62 seconds)

Search: [ ]

detectedoEn

[http://example.org/ontologia#empleoPublico\\_51](http://example.org/ontologia#empleoPublico_51)

## 5. Detecta registros con formato y tipo incorrecto.

SPARQL: Formato o tipo incorrecto del campo CP para Registros.

Ontop SPARQL endpoint		
Query 18 X	Query 16 X	Query 17 X
Query 15 X	Query 18 X	Query 17 X
Query 16 X	Query 15 X	Query 14 X
Query 17 X	Query 16 X	Query 15 X
Query 18 X	Query 17 X	Query 14 X
Query 19 X	Query 18 X	Query 15 X
Query 20 X	Query 19 X	Query 16 X
Query 21 X	Query 20 X	Query 17 X
Query 22 X	Query 21 X	Query 18 X
Query 23 X	Query 22 X	Query 19 X
Query 24 X	Query 23 X	Query 20 X
Query 25 X	Query 24 X	Query 21 X
Query 26 X	Query 25 X	Query 22 X
Query 27 X	Query 26 X	Query 23 X
Query 28 X	Query 27 X	Query 24 X
Query 29 X	Query 28 X	Query 25 X
Query 30 X	Query 29 X	Query 26 X
Query 31 X	Query 30 X	Query 27 X

```

1 PREFIX : <http://example.org/ontologia#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?error ?descripcion ?fecha ?uriRegistro
5 WHERE {
6   ?error a :ErrorFormato ;
7   ?errorDescripcion ?descripcion ;
8   ?errorFechaDetec ?fecha ;
9   ?errorDetectadoEn ?uriRegistro .
10 }
11

```

Showing 1 to 1 of 1 entries (in 0.015 seconds)

error	descripcion	fecha	uriRegistro
1 http://example.org/ontologia#error_registroFormato_206	El código postal contiene caracteres no numéricos	"2025-05-01T00:00:00"^^xsd:dateTime	<a href="http://example.org/ontologia#registro_206">http://example.org/ontologia#registro_206</a>

Showing 1 to 1 of 1 entries (in 0.015 seconds)

## Ontop SPARQL endpoint

Query 18 X	Query 16 X	Query 17 X	Query 15
Query 37 X	Query 36 X	Query 35 X	Query 34
<b>1</b> PREFIX : < <a href="http://example.org/ontologia#">http://example.org/ontologia#</a> >			
<b>2</b> PREFIX xsd: < <a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a> >			
<b>3</b>			
<b>4</b> SELECT ?error ?descripcion ?fecha ?uriRegistro			
<b>5</b> WHERE {			
?error a :ErrorFormato ;			
:errorDescripcion ?descripcion ;			
:errorFechaDetec ?fecha ;			
:errorDetectadoEn ?uriRegistro .			
<b>10</b> }			
<b>11</b>			

Showing 1 to 1 of 1 entries (in 0.015 seconds)		
error	descripcion	
1 <a href="http://example.org/ontologia#error_registroFormato_206">http://example.org/ontologia#error_registroFormato_206</a>	El código postal contiene caracteres no numéricos	

Showing 1 to 1 of 1 entries (in 0.015 seconds)

		Search: <input type="text"/>
fecha		uriRegistro
"2025-05-01T00:00:00""xsd:dateTime		<a href="http://example.org/ontologia#registro_206">http://example.org/ontologia#registro_206</a>

## 6. Detecta registros duplicados.

SPARQL: DNI duplicados de personas con distinto identificador.

Ontop SPARQL endpoint

```

Query 19 X Query 16 X Query 17 X Query 15 X Query 14 X Query 4 X Query 13 X Query 12 X Query 11 X Query 10 X Query 9 X Query 8 X Query 6 X Query 7 X Query 1 X Query 3 X Query 2 X Query 5 X Query 26 X Query 38 X
Query 39 X Query 37 X Query 36 X Query 35 X Query 34 X Query 33 X Query 32 X Query 31 X Query 27 X Query 25 X Query 24 X Query 23 X Query 22 X Query 19 X Query 28 X Query 21 X Query 26 X Query 29 X Query 30 X +
```

```

1 PREFIX : <http://example.org/ontologia#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?codigo ?descripcion ?fecha ?persona
5 WHERE {
6   ?error a :ErrorDuplicidad ;
7   ?errorDescripcion ?descripcion ;
8   ?errorFechaBucle ?fecha ;
9   ?errorDetectadoEn ?persona .
10
11 BIND(REPLACE(STR(?error), "http://example.org/ontologia#error_personaDuplicada_", "") AS ?codigo)
12
13 }
```

Table	Response	Pivot Table	Google Chart	Geo	Rows	</>
Showing 1 to 3 of 3 entries (in 0.511 seconds)						
codigo	descripcion	fecha	persona			
1	Identificación duplicada en el sistema	"2025-05-01T00:00:00""xsd:dateTime	<a href="http://example.org/ontologia#persona_1">http://example.org/ontologia#persona_1</a>			
201	Identificación duplicada en el sistema	"2025-05-01T00:00:00""xsd:dateTime	<a href="http://example.org/ontologia#persona_201">http://example.org/ontologia#persona_201</a>			
202	Identificación duplicada en el sistema	"2025-05-01T00:00:00""xsd:dateTime	<a href="http://example.org/ontologia#persona_202">http://example.org/ontologia#persona_202</a>			

Search:  Show 50 entries

## Ontop SPARQL endpoint

```

Query 18 X Query 16 X Query 17 X Query 15 X Query 14 X Query 4 X Query 13 X Query 12 X
Query 38 X Query 37 X Query 36 X Query 35 X Query 34 X Query 32 X Query 27 X Query 25 X
1 PREFIX : <http://example.org/ontologia#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?codigo ?descripcion ?fecha ?persona
5 WHERE {
6   ?error a :ErrorDuplicidad ;
7   :errorDescripcion ?descripcion ;
8   :errorFechaBucle ?fecha ;
9   :errorDetectadoEn ?persona .
10
11 BIND(REPLACE(STR(?error), "http://example.org/ontologia#error_personaDuplicada_", "") AS ?codigo)
12
13 }
```

			Table	Response	Pivot Table	Google Chart	Geo	</>
Showing 1 to 3 of 3 entries (in 0.511 seconds)								
codigo	descripcion	fecha						
Showing 1 to 3 of 3 entries (in 0.511 seconds)								
1	Identificación duplicada en el sistema	"2025-05-01T00:00:00""xsd:dateTime						
201	Identificación duplicada en el sistema	"2025-05-01T00:00:00""xsd:dateTime						
202	Identificación duplicada en el sistema	"2025-05-01T00:00:00""xsd:dateTime						

		persona
		<a href="http://example.org/ontologia#persona_1">http://example.org/ontologia#persona_1</a>
		<a href="http://example.org/ontologia#persona_201">http://example.org/ontologia#persona_201</a>
		<a href="http://example.org/ontologia#persona_202">http://example.org/ontologia#persona_202</a>



Cada consulta se ha evaluado en función de los siguientes criterios:

Tipo error	Entidad afectada	Relevancia	Exactitud	Tiempo Respuesta (s)	Cobertura	Trazabilidad Semántica	Reutilización	Observaciones
Error de asignación	TramiteServicio	Sí	Trámite 101	0.126	Detecta trámites sin departamento	Sí	Sí	Implementado con 4 mapeos modulares
Error de relación ausente	Personas	Sí	Persona 34	0.051	Detecta personas sin unidad asignada	Sí	Sí	
Error de consistencia temporal	FaseInicio	Sí	Fase Inicio 20	0.214	Detecta fechas fin anteriores a inicio	Sí	Sí	
Error de datos faltantes	EmpleoPublico	Sí	Empleo público 51	0.62	Detecta campos nulos obligatorios	Sí	Sí	
Error de tipo o formato	Registros	Sí	Registro 301	0.015	Detecta campos numéricos mal formateados	Sí	Sí	Usa REGEXP_LIKE para validar formato numérico
Error de duplicidad	Personas	Sí	Personas 1, 201 y 202	0.511	Detecta duplicados por identificación	Sí	Sí	Se requiere subconsulta para contar duplicados

## Anexo VIII. Evaluación “Consultas personalizadas”

Este anexo recoge los resultados completos obtenidos durante el proceso de validación de la funcionalidad de consultas personalizadas en lenguaje natural. El objetivo de esta funcionalidad es permitir que el usuario realice peticiones de información en lenguaje natural, las cuales son transformadas automáticamente en consultas SPARQL y ejecutadas sobre la base de datos Oracle, con posibilidad de exportar los resultados en formato Excel.

Para validar su funcionamiento, se han diseñado y evaluado múltiples consultas simuladas, representativos de necesidades dentro del contexto de atención al ciudadano. Cada consulta ha sido valorada según una serie de criterios definidos previamente, que incluyen la relevancia y exactitud de los resultados, el tiempo de respuesta, la necesidad de reformulación y la calidad del formato de salida.

En las tablas siguientes se documenta, para cada consulta, la petición original del usuario, la consulta SPARQL generada automáticamente, los resultados obtenidos, y una valoración detallada según los criterios establecidos. Esta evaluación permite identificar el grado de precisión semántica alcanzado por el sistema, así como su utilidad práctica.

**Primera y segunda iteración cálculo de tiempo respuesta (s):**

Consulta	¿Relevante?	Exactitud	Tiempo Respuesta (s)	Formato resultado	¿Reformulación?	Nivel Dificultad	Sección Evaluada	Observaciones
Muéstrame un listado de todas las personas registradas en el sistema que incluya el DNI.	Sí	Sí	1.62 0.03	Descarga fichero Excel	No	Básica	Personas	Resultados correctos y claros.
Muéstrame un listado de personas que pertenecen al departamento con código 101, incluyendo su nombre, apellidos y cargo.	Sí	Sí	1.76 0.02	Descarga fichero Excel	No	Media	Personas	Resultados correctos y claros. Incluye condición por relación (perPerteneceADep)
Muéstrame un listado de personas asignadas a un registro que pertenezca a la provincia de Valencia, incluyendo su nombre, apellidos, cargo y municipio.	Sí	Sí	1.91 0.03	Descarga fichero Excel	No	Avanzada	Personas	Relación entre dos entidades: Persona y Registro.

Listado de trámites y servicios con fase de iniciación. Tiene que tener la siguiente información: descripción y objetivo del trámite y servicios, objeto de la fase inicio y fecha inicio de la fase inicio.	Sí	Sí	2.18 0.06	Descarga fichero Excel	Sí	Avanzada	Trámites y Fase Inicio	
Trámites y servicios con fase de inicio, mostrando descripción, objetivo, fecha de inicio y requisitos	Sí	Sí	1.69 0.02	Descarga fichero Excel	No	Media	Trámites y Fase Inicio	Resultados correctos y claros.
Trámites y servicios con fase de instrucción, mostrando criterios de valoración, departamento tramitador y normativa	Sí	Sí	1.70 0.02	Descarga fichero Excel	No	Media	Trámites y Fase Instrucción	Resultados correctos y claros.
Subfase de Alegación	Sí	Sí	1.55 0.02	Descarga fichero Excel	No	Básica	Subfase Alegación	Resultados correctos y claros.

con trámite asociado: objeto, plazo y forma de presentación								
Trámites y servicios con subfase de justificación y cobro: objeto, fecha inicio y forma de presentación	Sí	Sí	2.12 0.02	Descarga fichero Excel	No	Avanzada	Subfase Justificación y Cobro	Resultados correctos y claros.
Trámites y servicios cuya fase final tenga recurso y sanción	Sí	Sí	2.05 0.02	Descarga fichero Excel	No	Media	Trámites y Fase Final	Resultados correctos y claros.
Listado de convocatorias de empleo público que pertenezcan al departamento 101, incluyendo descripción, número de plazas, grupo profesional y fecha de creación.	Sí	Sí	2.21 0.03	Descarga fichero Excel	Sí	Avanzada	Empleo público	Se añaden instrucción en el prompt con más detalle para que no confunda la relación.
Convocatorias de empleo público que requieren	Sí	Sí	1.48 0.02	Descarga fichero Excel	No	Avanzada	Empleo público	Resultados correctos y claros.

titulación específica y tienen el plazo abierto. Mostrar: descripción, tipo de personal, titulación, estado del plazo.								
Listado de etapas de convocatorias cuya fase actual esté en "Fase 1". Mostrar: convocatoria de empleo público, nombre etapa, fase, fecha de apertura, enlace.	Sí	Sí			Sí	Avanzada	Empleo público	No la resuelve.
Listado de convocatorias con pruebas selectivas tipo " Prueba teórica y práctica ", que tengan enlace de información y reducción de tasas.	Sí	Sí	2.21 0.03	Descarga fichero Excel	Sí	Avanzada	Empleo público y etapa	Resultados correctos y claros.

Mostrar: código, descripción, enlace, reducción tasa.								
Muéstrame un listado de convocatoria s de empleo público del departament o 154 que estén en la etapa actual fase 3, incluyendo la descripción de la convocatoria , el número de plazas y la fecha de publicación de la etapa.					Sí	Avanzada	Empleo público y etapa	

Nota: Es fundamental mantener la ontología sin instancias concretas (ABox), ya que pueden interferir en los resultados devueltos al mezclarse con los datos reales almacenados en la base de datos Oracle



# Referencias

- Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., & Auer, S. (2016).**  
*Quality assessment for Linked Data: A survey.* Semantic Web, 7(1), 63–93.  
<https://doi.org/10.3233/SW-150175>
- Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., & Zaveri, A. (2014).**  
*Test-driven evaluation of Linked Data quality.* In Proceedings of the 23rd International Conference on World Wide Web (WWW '14) (pp. 747–758). ACM.  
<https://doi.org/10.1145/2566486.2568002>
- Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodríguez-Muro, M., & Xiao, G. (2017).**  
*Ontology-based data access: A survey.* In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2017) (pp. 5511–5519).  
<https://www.ijcai.org/proceedings/2018/777>
- Tupayachi, J., Xu, H., Omitaomu, O. A., Camur, M. C., Sharmin, A., & Li, X. (2024).**  
*Towards next-generation urban decision support systems through AI-powered construction of scientific ontology using large language models—A case in optimizing intermodal freight transportation.* Smart Cities, 7(5), 2392–2421.  
<https://doi.org/10.3390/smartcities7050094>
- Firmenich, S. (2023).**  
*Interfaz conversacional para la recolección de datos en la Web Semántica* [Tesis de grado, Universidad Nacional de La Plata]. SEDICI.  
<https://sedici.unlp.edu.ar/handle/10915/166087sedici.unlp.edu.ar>
- Arteaga Meléndez, D. M. (2023).**  
*Diseño de un modelo explicativo basado en ontologías aplicado a un chatbot conversacional* [Tesis de licenciatura, Pontificia Universidad Católica del Perú]. PUCP.  
<https://tesis.pucp.edu.pe/bitstreams/bf554b9e-0820-4706-bfe4-6fd60c259415/download>

**Ciancio, M. (2024).**

*Creación de un prototipo de chatbot que permita interactuar con la historia del Ecuador registrada en periódicos antiguos [Tesis de grado, Universidad de Cuenca]. Repositorio Digital de la Universidad de Cuenca.*

<https://dspace.ucuenca.edu.ec/items/b964312f-8bfd-448c-be80-154c0d2bc1e9>

**Celi-Párraga, R. J., Varela-Tapia, E. A., Acosta-Guzmán, I. L., & Montaño-Pulzara, N. R. (2021).**

*Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional textual. AlfaPublicaciones, 3(4.1), 40–52. Retrieved February 9, 2025, from <https://alfapublicaciones.com/index.php/alfapublicaciones/article/view/123/373>*

**Lagos Ortiz, K. A. (2020).**

*Sistema de ayuda a la decisión basado en ontologías para el diagnóstico y prevención de las enfermedades en cultivos [Tesis doctoral, Universidad de Murcia]. Digitum.*

<http://hdl.handle.net/10201/94672>

**Paneque Romero, M. (2022).**

*Explotación de la semántica de dominio orientada a la integración, estandarización y análisis de datos [Tesis doctoral, Universidad de Málaga]. RIUMA.*

<https://www.educacion.gob.es/teseo/imprimirFicheroTesis.do?idFichero=CIUbg0Zo7Z0%3D>

**Gómez de Agüero Muñoz, J. (2024).**

*MethoOntoChat: Asistente conversacional del proceso metodológico de creación de ontologías basado en modelos del lenguaje [Trabajo Fin de Máster, Universidad Politécnica de Madrid]. Archivo Digital UPM.*

[https://oa.upm.es/83405/1/TFM\\_JAVIER\\_GOMEZ\\_DE\\_AGUERO\\_MUNOZ.pdf](https://oa.upm.es/83405/1/TFM_JAVIER_GOMEZ_DE_AGUERO_MUNOZ.pdf)

**Huaranga Junco, E. J. (n.d.).**

*Distilling ontologies from large language models [Trabajo Fin de Máster, Universidad Politécnica de Madrid]. Repositorio Digital UPM.*

[https://oa.upm.es/76086/1/TFM\\_EDGAR\\_JESUS\\_HUARANGA\\_JUNCO.pdf](https://oa.upm.es/76086/1/TFM_EDGAR_JESUS_HUARANGA_JUNCO.pdf)

**Brito Karadjian, B. J. (2024).**

*Sistema de generación de consultas SQL basado en peticiones de lenguaje natural a partir de un chat-bot* [Trabajo Fin de Grado, Universidad Europea de Valencia].

TITULA. <https://hdl.handle.net/20.500.12880/8877>

**Martínez, A. (2022).**

*Revisión de los chatbots basados en inteligencia artificial en la administración pública: Hacia una arquitectura para el gobierno.* Espacios Públicos, 25(1), 45–60.

<https://espaciosepublicos.uaemex.mx/article/view/21317>

**Suárez-Figueroa, M. C., & Gómez-Pérez, A. (2012).**

*NeOn Methodology for Building Ontology Networks: A Scenario-based Approach.* In M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, & A. Gangemi (Eds.), *Ontology Engineering in a Networked World* (pp. 9–34). Springer.

[https://doi.org/10.1007/978-3-642-24794-1\\_2](https://doi.org/10.1007/978-3-642-24794-1_2)

**RDF (Resource Description Framework)**

*World Wide Web Consortium (W3C).* (2014). *RDF 1.1 Concepts and Abstract Syntax.*

<https://www.w3.org/TR/rdf11-concepts/>

**OWL (Web Ontology Language)**

*W3C OWL Working Group.* (2012). *OWL 2 Web Ontology Language Document Overview (Second Edition).* World Wide Web Consortium (W3C).

<https://www.w3.org/TR/owl2-overview/>

**Musen, M. A. (2015).** *The Protégé project: A look back and a look forward.* AI Matters, 1(4), 4–12. <https://doi.org/10.1145/2757001.2757003>